Fast fruit calorie estimation using Faster R-CNN

Alan Samuel Aguirre Salazar
Faculty of Engineering.
Universidad Panamericana
Jalisco, México
Emails: ASASauqui@protonmail.com / 0227221@up.edu.mx

***Abstract*— In today's world, obesity has become a global health problem that affects millions of people. Access to high-calorie foods and lack of physical activity are main factors in its increase. In order to address this problem, the use of artificial intelligence (AI) has been proposed to estimate the caloric content of food. In this particular context, an algorithm has been developed that uses convolutional neural networks (CNN) for the detection and classification of fruits and image processing techniques to extract the volume and surface area of the fruit from its image. The proposed method involves pre-processing the input images, applying Faster R-CNN for fruit detection and classification, and then extracting the fruit volume by segmenting the fruit from the image and estimating. of its dimensions with the help of a reference object. The main result of this study is an approximation of the caloric content of 3 different types of fruits, which provides a useful tool for meal planning and diet management. Furthermore, this method could be scaled up to other types of food, which could have an impact in the fight against obesity and other diet-related diseases.**

Keywords—Calorie estimation, Convolutional Neural Networks, Faster R-CNN, image processing.

I. Introduction

The article addresses a current need in society, where the problem of obesity and poor nutrition is increasingly important. Since 1980, the prevalence of obesity in American adults has skyrocketed from 14% to 42% [1]. Although there is no clear answer to this epidemic, it is accepted that some of the causes that drive it (related to food) are the high availability of processed foods that are high in calories, fats, and added sugars, as well as the increase in portion sizes and lack of access to healthy foods [1].

Knowing what we consume and what it contains is important to have a balanced diet. The approach proposed in the article can be an effective and efficient solution to estimate the caloric content of fruits and promote healthy eating habits in the population. This article presents an algorithm to accurately estimate the caloric content of 3 types of fruits using Faster R-CNN and image processing techniques [2]. The proposed method involves the pre-processing of the input images, the use of Faster R-CNN for the detection and classification of the fruit, and the extraction of the fruit volume by segmenting the fruit from the image and estimating its dimensions with the help of a reference (in this case, a coin).

This type of proposal using CNN for food detection has already been addressed in various works using various architectures [3]. The integration of AI techniques, such as CNN and image processing, provides a promising avenue for estimating the caloric content of foods accurately. The concept of using deep residual learning in image recognition has demonstrated remarkable results in various applications [2]. Additionally, large-scale visual recognition challenges, such as the ImageNet challenge, have showcased the potential of image recognition and classification algorithms [3].

By leveraging these advancements in the field of computer vision, the algorithm proposed in this article aims to provide a practical solution for estimating the caloric content of fruits. By focusing on fruits initially, this method can serve as a useful tool for meal planning and diet management. Furthermore, the scalability of this approach to other types of food could have a significant impact in the fight against obesity and other diet-related diseases.

II. . State of art

Previous works related to the resolution of this problem began first with basic machine learning techniques, but in recent years more works have come out that address it using Deep Learning, but each of these using different architectures and methodologies for caloric estimation. but in turn, most use CNN for the detection and classification of food. Some of these works do not present a defined CNN architecture, since they focus more on the algorithm for calorie estimation, leaving the use of CNN for detection in a secondary part.

Pouladzadeh et al. [4], did not define the CNN architecture exactly in their proposal, but general details of how it is made up are mentioned; A deep neural network with a CNN layer was used to train the pre-trained model with a set of labeled positive images and a set of negative images (of the foods to be identified). The system used a rectified linear unit (ReLU) activation function to model the output of the neuron. They wanted to find the set of weights and biases that make the cost as small as possible, for this the stochastic gradient descent algorithm was used to find a set of weights and biases that minimize the quadratic cost function, with this they did the job of sorting out the food. For the calorie estimation task, an individual photo of the food was simply used together with a reference object whose volume is already known (a thumb in this case), and also, for comparison, a calculation of calories by estimating the distance thanks to the use of a mobile device that contains an accelerometer and magnetic field sensors. Satisfactory results were obtained both in the detection of foods and in the estimation of their calories, with 99.9% accuracy in the detection of foods and standard deviations of less than one unit of calories.

Kasyap et al. [5], made use of an API for the detection of objects, from Tensorflow and CNN was also used for the classification of the food, although the architecture is not specified exactly, it is mentioned that it was built using sequential, convolutional, pooling and full connection. The accuracy of this proposal was 92%, achieved thanks to the use of its activation function, the optimizer and the initializer, of these three only the optimizer is specified, which was the root mean square propagation (RMSprop). To estimate the calories, 2 reference points or perspectives (images) were used to calculate the volume of the object: a top view of the food and a side view; use was also made of a reference object whose volume is known to be able to determine the calories of the food in question (a thumb again).

Okamoto et al. [6], propose the use of CNN with the "Network in Network" (NIN) architecture, which introduces a new layer known as mlpconv layer, which uses 1x1 size filters that act as MLP units that can learn any linear function. This architecture is made up of: convolutional layers, followed by a ReLU activation layer; the mlpconv layer; global pooling layers and the softmax layer. And for the estimation of calories, a 3D reconstruction was used, requiring, unlike other works, a single point of view; Food serving height was assumed to be related to food size and food categories, and the quadratic curve of 2D food size to its calories was estimated. The actual size of the food regions is estimated based on the size of the reference object, allowing the actual size of the food to be obtained using an equation.

Ege et al. [7], propose the use of multi-job learning with CNN, where the image is given as input value, and the image category and its calorie estimate are given as output. The architecture for this multi-task CNN was based on VGG16, which is popular for its simplicity and ability to achieve high performance in a wide variety of computer vision tasks. No dropout was used here, but batch normalization for the fc6 layer and the fc7 layer. The fc6 layer is shared by both tasks and the fc7 layer is diverted to each task. Here, unlike the other works where other external techniques are used to calculate calories, this task was treated as a regression problem.

Liang et al. [8], proposed the use of the Faster R-CNN architecture (which is an evolution of the R-CNN architecture and is considered as one of the most accurate approaches for image object steering) to detect each food and classify it, and the calibration object (reference, which in this case is a coin). Its sequence flow to make your proposal is divided into 5 steps: image acquisition, object detection, image segmentation, volume estimation and calorie estimation.

As can be seen, all these works use CNN for the detection and classification of food, with different architectures and methodologies for calculating calories. If a 3D reconstruction or sensors are not used for the exact calculation, it is best to use reference objects whose precise volume is already known, in this case, the best is a coin and not so much a thumb, since the thumb can vary between users, and it is even a little more complicated to calculate its volume, on the other hand, a coin is more constant in terms of volume and its calculation is easier.

III. Proposed Methodology

The proposed methodology combines the use of a Faster R-CNN for object classification and detection, along with manual image segmentation using conventional techniques for calorie estimation. The following points outline the key steps involved in the methodology:

1. **Utilizing Faster R-CNN for Object Classification and Detection:** The Faster R-CNN (Region-based Convolutional Neural Network) architecture is chosen for its superior performance in object detection tasks [9]. It consists of two modules: a Region Proposal Network (RPN) for generating region proposals and a Fast R-CNN network for classifying and refining these proposals. By leveraging the power of Faster R-CNN, accurate and efficient fruit classification can be achieved.
2. **Manual Image Segmentation for Calorie Estimation:** The estimation of calories is performed through manual image segmentation using conventional techniques. The input fruit images are preprocessed to enhance their quality and remove any potential noise or artifacts. Then, a thresholding technique is applied to segment the fruit region from the background, followed by contour detection to extract the boundary of the fruit [10]. This manual segmentation approach allows for precise estimation of the fruit's dimensions, which are crucial for calorie estimation.

By combining the capabilities of Faster R-CNN for object classification and detection and manual image segmentation for calorie estimation, the proposed methodology aims to provide an accurate and efficient system for estimating the caloric content of fruits.

A. *Faster R-CNN.*

Faster R-CNN (Region-based Convolutional Neural Networks) is a popular object detection framework that achieves high accuracy and efficiency. It combines two key components: a region proposal network (RPN) and a convolutional neural network (CNN) for object detection.

The RPN generates region proposals, which are potential object bounding boxes, by sliding a small network over the convolutional feature map. These proposals are then refined using bounding box regression and non-maximum suppression to filter out redundant detections. The RPN is trained end-to-end with the rest of the Faster R-CNN model.

The CNN, often based on popular architectures like ResNet or VGGNet, serves as the backbone for feature extraction. It takes an image as input and produces a convolutional feature map that captures the spatial information of the image. This feature map is then fed into the RPN to generate region proposals and also shared with the region of interest (RoI) pooling layer for subsequent object classification and bounding box regression.

The Faster R-CNN model used in the object detection framework incorporates the Faster R-CNN ResNet-50 architecture as the backbone for feature extraction. Faster R-CNN ResNet-50 is a convolutional neural network (CNN) model that has demonstrated excellent performance in image recognition tasks. The use of a pre-trained Faster R-CNN ResNet-50 model in the Faster R-CNN framework provides advantages such as leveraging pre-learned features and reducing training time.

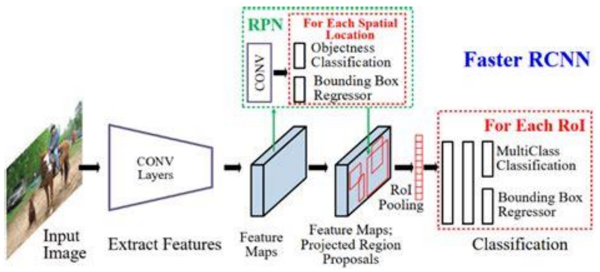

Illustration 1: Faster R-CNN FPN pipeline.

The architecture used in the Faster R-CNN model consists of several components:

1. **Backbone Network:** The backbone network, represented by Faster R-CNN ResNet-50 consists of multiple convolutional layers followed by pooling layers, which downsample the spatial dimensions while increasing the number of channels. This process captures both low-level and high-level features in the image [11].

   Mathematically, the backbone network can be represented as:

   $$\mathit{feature\_map = Backbone(input\_image)}$$

   where feature_map represents the output feature map obtained from the backbone network.

2. **Region Proposal Network (RPN):** The RPN takes the feature map from the backbone network and generates region proposals by predicting bounding box coordinates and objectness scores for each anchor box. The anchor boxes are predefined boxes of different scales and aspect ratios that act as reference points. The RPN is trained to learn how to adjust the anchor boxes to tightly fit the object instances [11].

   Mathematically, the RPN takes the convolutional feature map as input and predicts the bounding box coordinates (Δx, Δy, Δw, Δh) and objectness scores (p) for each anchor box. This can be represented as:

   $$\Delta x, \Delta y, \Delta w, \Delta h, p = \mathit{RPN(feature\_map)}$$

   where Δx, Δy represent the adjustments to the anchor box coordinates, Δw, Δh represent the adjustments to the width and height of the anchor box, and p represents the objectness score.

3. **RoI Pooling Layer:** After obtaining the region proposals from the RPN, the RoI pooling layer extracts fixed-size feature maps from the backbone feature map for each proposal. These fixed-size feature maps are then fed into a fully connected network for object classification and bounding box regression.

   The RoI pooling process involves dividing each region proposal into equal-sized sections and applying a pooling operation within each section to obtain a fixed-size feature map. This allows the subsequent network layers to operate on inputs of consistent sizes.

   Mathematically, the RoI pooling layer can be represented as:

   $$\mathit{fixed\_size\_feature\_maps = RoIPooling(feature\_map, region\_proposals)}$$

   where fixed_size_feature_maps represent the fixed-size feature maps obtained by applying RoI pooling on the shared backbone feature map using the region proposals.

These components work together in an end-to-end manner, allowing Faster R-CNN to accurately detect objects.

B. *Dataset.*

The availability of a pre-existing dataset in the desired format (VOC XML, CSV, or YOLO) specifically tailored to the classes of fruits and coins was limited. Despite an extensive search, only one dataset was found, namely "Fruit Images for Object Detection" created by MUHAMMEDBUYUKKINACI. This dataset served as the foundation for our research, but it required further augmentation to incorporate additional images and corresponding VOC XML annotations.

The dataset, although valuable, remained relatively small due to the limited availability of labeled data for fruit and coin object detection. This presents an area of opportunity for future research to expand the dataset by including a more diverse range of fruit and coin images, potentially leveraging crowd-sourcing or expert annotation. By increasing the dataset size, both in terms of the number of images and the variability of the captured instances, the model's performance and generalization capabilities can be enhanced.

Illustration 2: Kind of images in the dataset

C. *Training.*

The training parameters were selected based on their specific functionalities and their impact on the training process. These parameters were chosen to optimize the performance and convergence of the model.

- The number of epochs determines the number of complete passes through the training dataset during training. In this case, the model was trained for 20 epochs to allow sufficient iterations for the model to learn and converge.
- The num_workers parameter, set to 4, controls the number of subprocesses used for data loading during training. By utilizing multiple workers, the data loading process can be parallelized, resulting in faster training.
- The learning_rate parameter, set to 0.001, determines the step size at which the model learns during training. A smaller learning rate was chosen to ensure a more cautious and precise update of the model's parameters.
- The batch_size parameter determines the number of samples processed in each iteration of training. With a batch size of 8, the model processed eight images simultaneously during each training iteration, allowing for efficient memory utilization and computational throughput.
- The step_size (5) and gamma (0.1) parameters are related to the learning rate schedule used during training. The step_size specifies the number of epochs after which the learning rate is reduced, while the gamma controls the reduction factor. This schedule helps the model to fine-tune its learning rate over time, enabling more accurate and stable convergence.
- The momentum parameter, set to 0.9, was chosen to accelerate the optimization process. By incorporating momentum, the model can overcome local minima and converge faster towards an optimal solution.
- The weight_decay parameter, set to 0.0005 [12], applies L2 regularization to the model's weights during training. This regularization helps prevent overfitting by penalizing large weight values and promoting smoother and more generalizable models.

Regarding the optimizer, a gradient descent optimizer was employed for updating the model's parameters based on the computed gradients. The optimizer efficiently adjusts the model's weights to minimize the loss function [13]. Additional details about gradient descent optimization can be found in the cited sources.

These training parameters and optimizer settings were chosen based on their effectiveness in optimizing the model's performance and achieving accurate object detection results.

D. *Image cropping.*

After making predictions on the input images, the next step in the pipeline is to crop the images based on the predicted bounding boxes [14]. This process involves extracting the regions of interest from the original image, which contain the objects of interest identified by the object detection model.

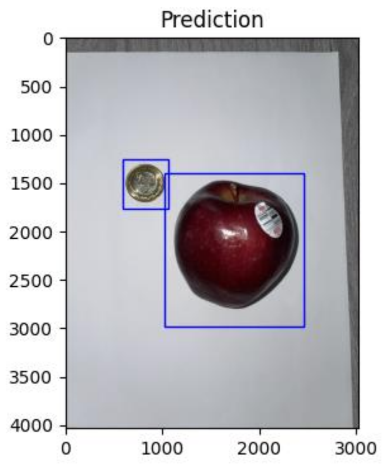

Illustration 3: Prediction and bounding boxes.

The provided code snippet demonstrates a function called get_cropped_image that performs the image cropping operation. Let's understand the steps involved:

1. **Image Copy:** The original image is copied to ensure that the cropping operation does not modify the original image. This step is essential to maintain the integrity of the input data.

2. **List Initialization:** A list named cropped_images is initialized [15]. This list will store the cropped images along with their corresponding labels.

3. **Cropping:** For each predicted bounding box in the nms_prediction_cpu['boxes'] list, the code performs the following steps:

   a. The image is cropped using the coordinates of the bounding box [16]. The box_i variable represents the bounding box coordinates in the format [x_min, y_min, x_max, y_max].

b. The image is cropped using the specified bounding box coordinates: img_copy[:, int(box_i[1]) : int(box_i[14]), int(box_i[0]) : int(box_i[2])].

c. The cropped image is converted from a PyTorch tensor to a PIL image using a function called tensor_to_PIL_image.

d. The PIL image is converted into a numpy array using np.array(crop_img).

e. The cropped image and its corresponding label are appended to the cropped_images list.

4. **Output:** Finally, the function returns the cropped_images list, which contains the cropped images and their respective labels.

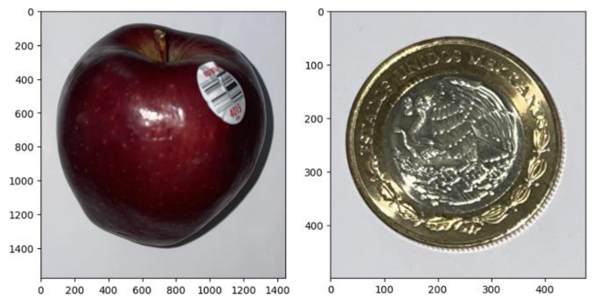
Illustration 4: Cropped images.

E. Image segmentation and area calculation.

Image segmentation is the process of dividing an image into meaningful and semantically consistent regions [17]. It aims to identify and extract specific objects or regions of interest within an image. In the context of computer vision, segmentation plays a crucial role in various applications, including object recognition, scene understanding, and image editing.

One approach to image segmentation involves the use of contour detection and area calculation to extract regions of interest [18]. The code snippet provided demonstrates a method called get_area_percentage that performs segmentation and calculates the area percentage of each segmented region compared to the original image. Let's walk through the code step by step:

1. **Image Preprocessing:** The original image and a list of cropped images are passed as inputs to the get_area_percentage function. The dimensions of the original image are extracted to calculate the area percentages later on.

2. **Contour Detection:** For each cropped image in the list, the code performs the following steps:

a. The cropped image is copied and converted from RGB to grayscale using OpenCV.

b. A binary inverse threshold is applied to the grayscale, resulting in a binary image with foreground objects as white and the background as black.

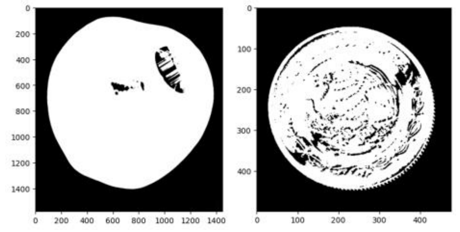
Illustration 5: Threshold result.

c. The contours of the objects are extracted using the OpenCV library. The contour retrieval mode is set to cv2.RETR_TREE, which retrieves all the contours and reconstructs the full hierarchy of nested contours [19]. This allows for more precise analysis of the object structure. The contour approximation method is specified as cv2.CHAIN_APPROX_SIMPLE, which compresses horizontal, vertical, and diagonal segments of the contours, resulting in a more compact representation [19]. This helps reduce the memory requirements and computational complexity while retaining the essential shape information of the objects.

d. Next is find the largest contour based on the contour area.

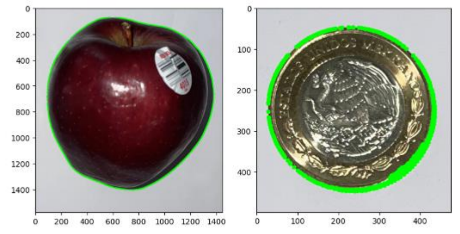
Illustration 6: Contour of the images.

3. **Area Calculation:** After obtaining the contour of the object, the code calculates the area of the contour using the cv2.contourArea function. This function calculates the area enclosed by the contour, providing a measure of the segmented region's size in terms of the number of pixels it occupies. By calculating the area, the code obtains a quantitative representation of the object's extent, which can be useful for various applications such as object classification or size-based analysis.

4. **Percentage Calculation:** The code calculates the percentage of the area covered by the contour relative to the size of the cropped image using contour_area / (height_i * width_i). Additionally, it calculates the percentage of the segmented region's area compared to the original image by multiplying the contour

percentage with the cropped image percentage obtained earlier.

5. **Result Aggregation**: The code distinguishes between the segmented objects labeled as "coin" and others labeled as "fruits." The area percentages are stored accordingly in the variables coin and fruits.

6. **Output:** Finally, the function returns the area percentage of the "coin" object and a list of tuples containing the labels and area percentages of the segmented "fruits" objects.

F. Image segmentation and area calculation.

The calculation of calories involves a series of steps, and each fruit type requires its own method to estimate volume. The provided code includes a Fruit class that contains specific information for different fruits, such as calorie content, width, and density. Additionally, there is a Coin class that provides the area of the coin for reference. The combined explanation of the calorie calculation process, along with the specific formulas and methods for volume estimation, is as follows:

To calculate the calories of each fruit, the code employs the following steps:

1. **Fruit Area Estimation:** The area of each fruit is determined based on the fruit_area_percentage, assuming a proportional relationship between the fruit area and the reference coin's area. The formula to calculate the fruit's area (fruit_area) is:

   fruit_area = (fruit_area_percentage * Coin().area) / coin [20]

2. **Fruit Volume Estimation:** The Fruit class provides a get_volume method that calculates the volume of each fruit based on its type and area. The specific formulas for volume estimation are defined separately for each fruit type. The code considers three fruit types: 'apple,' 'orange,' and 'banana.' The volume estimation formulas for each fruit are as follows:

   a. Apple and Orange:

      i. Radius Calculation:

      $$r = \sqrt{(area / \pi)}$$

      ii. Volume Calculation:

      $$volume = (4/3) * \pi * (r^3)$$

   b. Banana:

      i. Volume Calculation:

      $$volume = area * fruit\ width$$

   Note that the fruit width expression retrieves the specific factor value for the corresponding fruit type.

3. **Fruit Mass Estimation:** The mass of each fruit is calculated by multiplying the fruit's volume by its corresponding density. The Fruit class includes a density dictionary that holds density values for each fruit type. The formula for mass calculation is:

   Fruit mass = Fruit volume * Fruit density [20]

4. **Calorie Calculation:** The calorie content of each fruit is determined by multiplying its mass by the calorie value per 100 grams. The Fruit class contains a calories dictionary that stores the calorie values for each fruit type. The calorie calculation formula is:

   Fruit calories = (Fruit mass * Fruit calories per 100 grams) / 100 [20]

The above calculations are applied iteratively for each fruit in the fruits list, resulting in a list of calorie values.

It's important to note that the specific formulas and methods used for volume estimation and density calculation are defined within the Fruit class. These formulas may vary based on the fruit type and the available data.

In summary, the provided code implements a calorie calculation process based on area, volume, and mass estimation for various fruit types. These calculations are performed using specific formulas and data stored within the Fruit class.

IV. Results

In terms of the Intersection over Union (IoU) metric for bounding box detection, the model achieved an Average Precision (AP) of 0.475 when considering IoU values ranging from 0.50 to 0.95. For IoU = 0.50, the AP was 0.862, and for IoU = 0.75, the AP was 0.491. These metrics indicate that the model achieved moderate precision in accurately localizing and classifying objects.

When considering different object areas, the model achieved an AP of -1.000 for small objects, indicating that the performance in detecting small objects was poor. For medium-sized objects, the AP was 0.466, and for large objects, the AP was 0.491, indicating relatively better performance in detecting larger objects.

In terms of Average Recall (AR), which measures the proportion of ground-truth objects correctly detected, the model achieved an AR of 0.603 for maxDets=100. This indicates that the model was able to detect 60.3% of the objects on average across different IoU thresholds and object areas.

Overall, the model's performance can be considered decent, with reasonably good precision and recall for objects of larger sizes. However, there is room for improvement, particularly in detecting small objects.

Regarding the estimation of calories, multiple images of different fruits were loaded to predict their calorie content using the trained model. Subsequently, an algorithm was applied to calculate the calorie values. The obtained metrics for each fruit were as follows:

	Mean	**Median**	**STD**	**Max.**	**Min.**
Apple	106.36	105.94	5.13	116.49	100.52
Banana	170.74	169.24	4.33	178.61	165.08
Orange	302.55	126.88	393.91	1183.36	123.11

Table 1: Calories estimation table.

For apples, the mean calorie value was 106.36, with a median of 105.94 and a standard deviation of 5.13. The maximum calorie count observed was 116.49, and the minimum was 100.52.

For bananas, the mean calorie value was 170.74, with a median of 169.24 and a standard deviation of 4.33. The maximum calorie count observed was 178.61, and the minimum was 165.09.

For oranges, the mean calorie value was 302.55, with a median of 126.88. However, there was a significant outlier with a calorie count of 1183.36, which is unrealistic for an orange. This outlier can be attributed to a segmentation algorithm failure, where an excessively large area was considered, resulting in an erroneous calculation of calories for an oversized orange. It is important to note that the presence of this outlier impacts the overall statistics, including the mean and standard deviation.

Based on these results, it can be inferred that the model generally performed well in estimating the calorie content of apples and bananas. However, in the case of oranges, the presence of the outlier indicates the need for further refinement in the segmentation algorithm to avoid such inaccuracies.

V. CONCLUSION

In conclusion, the results of the Faster R-CNN model training indicate its potential for accurately detecting and classifying objects, although there is room for improvement. The model's performance could have benefited from a larger and more diverse dataset specifically tailored to the problem of fruit detection and calorie estimation. However, due to the lack of an existing suitable dataset, a new one had to be created almost from scratch, which may have limited the model's generalization ability.

Furthermore, the limitations of the image segmentation algorithm used in the calorie estimation process became apparent, particularly in the case of oranges where an outlier with an unrealistically high calorie count was observed. To address this, a more efficient and robust deep learning architecture can be proposed for improved image segmentation. One such architecture could be an instance segmentation model, such as Mask R-CNN, which combines object detection and pixel-level segmentation to precisely delineate the boundaries of each fruit instance within the image. This would potentially yield more accurate and reliable results in estimating the calorie content of fruits.

In summary, while the trained Faster R-CNN model showed promising results, there is room for further enhancement. A larger dataset specific to the problem domain and the adoption of more advanced deep learning architectures, such as Mask R-CNN, can contribute to achieving better performance in fruit detection and calorie estimation tasks.

REFERENCES

1. Dariush Mozaffarian, Perspective: Obesity—an unexplained epidemic, *The American Journal of Clinical Nutrition*, Volume 115, Issue 6, June 2022, Pages 1445–1450, https://doi.org/10.1093/ajcn/nqac075.
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90
3. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y
4. P. Pouladzadeh, P. Kuhad, S. V. B. Peddi, A. Yassine and S. Shirmohammadi, "Food calorie measurement using deep learning neural network," *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, Taipei, Taiwan, 2016, pp. 1-6, doi: 10.1109/I2MTC.2016.7520547.
5. V. B. Kasyap and N. Jayapandian, "Food Calorie Estimation using Convolutional Neural Network," *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, Coimbatore, India, 2021, pp. 666-670, doi: 10.1109/ICSPC51351.2021.9451812.
6. Okamoto, K., & Yanai, K. (2016, October). An automatic calorie estimation system of food images on a smartphone. In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management (pp. 63-70).
7. Ege, T., and Yanai, K. (2017, May). Simultaneous estimation of food categories and calories with multi-task CNN. In 2017 fifteenth IAPR international conference on machine vision applications (MVA) (pp. 198-201). IEEE.
8. Liang, Y., & Li, J. (2017). Deep learning-based food calorie estimation method in dietary assessment. arXiv preprint arXiv:1706.04062.
9. S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern

Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

10. Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics, (6), 610-621. doi: 10.1109/TSMC.1973.4309314.

11. R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

12. Andrew Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (ICML '04). Association for Computing Machinery, New York, NY, USA, 78. doi: 10.1145/1015330.1015435.

13. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

14. Liao, M., Shi, B., & Bai, X. (2013). A fast algorithm for multilevel thresholding. Journal of Software, 8(6), 1386-1393.

15. Malik, J., & Belongie, S. (2000). Contour and texture analysis for image segmentation. International Journal of Computer Vision, 43(1), 7-27.

16. Mori, G., & Malik, J. (2003). Estimating human body configurations using shape context matching. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), (pp. 786-793).

17. Arbeláez, P., Pont-Tuset, J., Barron, J. T., Marques, F., & Malik, J. (2014). Multiscale combinatorial grouping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (pp. 328-335).

18. Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5), 603-619.

19. Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66.

20. Ello-Martin, J. A., Roe, L. S., Ledikwe, J. H., Beach, A. M., & Rolls, B. J. (2007). Dietary energy density in the treatment of obesity: a year-long trial comparing 2 weight-loss diets. The American Journal of Clinical Nutrition, 85(6), 1465-1477. [Link: https://pubmed.ncbi.nlm.nih.gov/17556681/]