Comparative Analysis of Conventional and Deep Learning Algorithms for Apple Detection

Hrishit Das

hrishit.das.2024@bristol.ac.uk

Ruchita K. Vehale
Pranav Shirgur

Research Article

Keywords:

Posted Date: August 25th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7433552/v1

Additional Declarations: No competing interests reported.

Comparative Analysis of Conventional and Deep Learning Algorithms for Apple Detection

Hrishit Das, Ruchita K. Vehale, Pranav Shirgur
MSc Robotics, University of Bristol

Abstract

Precise apple counting is one of the vital tasks in the overall context of agricultural applications, from yield estimation to resource allocation and several other logistical issues about harvesting. The paper has discussed a comparison of conventional image processing techniques with modern deep learning for the detection of apples. Apple detection research, focusing on apples under difficult orchard situations using the MinneApple dataset to ensure strong deep learning-based detection due to YOLOv8, was studied. In particular, it compared traditional approaches using HSV color segmentation and morphological operations against a fine-tuned YOLOv8 model, improved by Roboflow. These studies proved the deep learning approach of difficult scenarios with very high precision, recall, and F1 score, thus enabling it to rightly distinguish apples on the trees from those lying on the ground. Results are highly useful in understanding how traditional and modern methods can be integrated into agricultural automation.

Introduction

Apple detection has been a critical task in modern agriculture for yield estimation, resource planning, and harvesting. Although the classical image-processing-based approaches, such as HSV color segmentation and texture-based methods, are highly efficient, they have many limitations in complex orchard environments due to occlusions, varying sizes of apples, and illumination changes. Therefore, in this paper, the machine learning-based approach was explored for apple detection using a YOLOv8 object detection model fine-tuned on the MinneApple dataset with high-resolution images and annotations. The paper compares traditional methods with YOLOv8 and evaluates their performances with an emphasis on the potential of integrating them. This work advances agricultural automation toward scalable and efficient real-world apple detection solutions.

Literature Review

Apple detection and segmentation are essential steps in many agricultural automation tasks, including yield estimation, quality assessment, and robotic harvesting. Traditional approaches, mostly comprising HSV color segmentation, morphological operations, and watershed algorithms, dominate due to their simplicity and computational efficiency [1, 2]. In most of the cases, HSV segmentation of apples separates the apples by splitting hue and intensity, while morphological operations together with circle fitting tune this result, especially when apples overlap [3].

These methods, however, have been wanting in more complicated scenarios, like changing lighting conditions or occlusions. The recent methods about deep learning, especially those based on the YOLO framework, solve the above problems much more adaptively and with more accuracy. YOLOv5s-BC enhances the baseline YOLOv5 with attention mechanisms and better feature pyramid networks, ensuring superior performance in the detection of apples under difficult conditions [4]. YOLOv8 further proves to be effective, with the ability to perform 3D shape fitting to deal with apples of variable size and occlusion overlap regions [5]. Generating synthetic data using OpenAI's DALL·E model was proved effective to train the YOLO model to bridge the gap between synthetic to real [6].

Hybrid methods combining traditional and deep learning approaches excel in tasks like detecting damaged apples, with YOLOv3's optimized anchor boxes achieving high precision in identifying bruising and decay.

Data Acquisition and Datasets

Effective apple counting in real-world applications requires robust imaging systems capable of handling diverse environmental conditions. The following types of image sensors/imaging systems can be employed.For this project, RGB cameras are assumed as the primary imaging system due to their affordability, ease of use, and compatibility with the MinneApple dataset.

The MinneApple dataset Authors: Haeni, Roy, and Isler, 2019 is a benchmark dataset created for the specific purposes of detecting and segmenting apples within orchard environments. It consists of high-resolution RGB images in natural conditions, with detailed annotations as bounding boxes and segmentation masks. This forms a very good ground to test the performance of deep learning models to be used for apple counting.

0.1 Quality

High-quality image capture, mostly of great resolution, comprises the MinneApple dataset so that in almost all pictures, there are small or partly occluded apples. Well-structured annotations are a must when it comes to training models for object detection, which surely contributes to its consistency. This would reduce possible errors when going through either the training or validation phase, thus making the dataset more reliable and fitted to be used in creating a strong model.

0.2 Variability

One of the key strengths of the MinneApple dataset is its variability. The dataset includes images captured under diverse environmental conditions, such as bright sunlight, partial shading, and overcast weather. This diversity reflects the real-world challenges encountered in orchards. Additionally, it features scenarios with apples at various stages of visibility, such as those partially hidden by leaves or branches, as well as apples fallen on the ground. This variability ensures that the dataset can support the development of models capable of generalizing to a wide range of conditions and scenarios.

0.3 Appropriateness

The MinneApple dataset is highly appropriate for this project due to its focus on apple detection in orchard environments. Its relevance to the project's objectives ensures that the dataset requires minimal adaptation for use. The inclusion of challenging scenarios, such as overlapping apples and apples on the ground, aligns directly with the goals of developing a robust and practical detection model. Furthermore, as a publicly available benchmark dataset, MinneApple allows for reproducibility and comparison with other studies, making it a dependable and practical choice.

In this project, the MinneApple dataset was used to train and validate a YOLOv8-based detection model. Its high quality, variability, and direct relevance to apple counting tasks ensured that it provided a strong foundation for developing a robust deep learning solution.

Methodology

0.4 Conventional Approach

The traditional approach followed here is based on deterministic, rule-based methods using image features of color, texture, and intensity. Some of the major approaches followed are listed below.

0.4.1 HSV Color Segmentation

The hue, or chromatic information, is separated from intensity in the HSV color space and is very effective for color-based segmentation. This model turns out to be very suitable for the detection of changes in colors of apples. The HSV color space separates color (hue) from intensity (value); as a result of this, it proves efficient in finding red hues under variable lighting conditions.

0.4.2 Morphological Operations

Morphological operations will be applied for refining the binary mask, removal of noise, and filling gaps to effectively segment the image. The presence of noise or small gaps in segmentation may reduce its accuracy. The morphological operations enhance the quality of the mask by refining the object boundaries. For morphological operations, a small elliptical-shaped kernel is used so that refinement of features is effective without over-smoothing.

0.4.3 Bilateral Filtering

The refined binary masks are further smoothed using bilateral filtering, which retains essential edges and hence preserves object boundaries. Unlike other smoothing techniques, bilateral filtering preserves edges while reducing noise, hence suitable for refining object segmentation.

Formula:

$$I_{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in S} G_s(\|x - x_i\|) G_r(|I(x) - I(x_i)|) I(x_i)$$

Where:

- G_s: Gaussian spatial weight
- G_r: Gaussian intensity difference weight
- W_p: Normalization factor

0.4.4 Distance Transform and Watershed Algorithm

The distance transform along with the watershed algorithm separates overlapping apples efficiently. It highlights the center of the objects as the peaks and applies the watershed algorithm on those peaks for efficient separation of the overlapping regions.
Process: It gives, for each pixel, its distance from the closest background pixel in the form of a gradient map—the object centers form the peak points of the distance maps.

$$d(p) = \min_{b \in B} \|p - b\|$$

Here, $d(p)$ stands for the distance of the given pixel from its nearest background pixel.By taking this set of peaks as markers for watershed-based flooding to construct basins in segments, the method eventually helps segregate out-of-focused overlapped regions.

$$D(x, y) = \min\{d((x, y), (x', y')) \mid (x', y') \in \text{background}\}$$

0.4.5 Local Binary Patterns (LBP)

LBP is a powerful texture-based feature extraction method that helps in improving the detection by gaining textural information. LBP extracts local features of texture that can go deep into knowing the characteristics of the apple's surface and help in classification or quality assessment.
Process:

1. The image is converted to grayscale.
2. The LBP code is computed by comparing the intensity of every pixel with its neighborhood.

$$\text{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} s(i_p - i_c) 2^p$$

Where:

- i_c: Intensity of the central pixel
- i_p: Intensity of the neighboring pixel
- $s(x) = 1$ if $x \geq 0$, otherwise $s(x) = 0$

Normalize the LBP image for better highlighting the textural features.

0.4.6 Canny Edge Detection

Canny edge detection and contour detection give outlines and define boundaries of apples present within an image. Well-defined boundaries are essential for detection and segmentation of apples with high accuracy, which is achieved by detecting and enhancing the edge details.

0.4.7 Contour Detection:

Contours are detected from the edges. The contours detected will be smoothed and simplified using algorithms like Douglas-Peucker.

0.5 Deep Learning Approach

This work counts the apples in an orchard environment using a deep learning-based approach. The proposed approach here is based on object detection using a pre-trained YOLOv8 model fine-tuned on the MinneApple dataset for more accuracy.

0.5.1 Data Acquisition and Preparation

Data acquisition begins with the MinneApple dataset, which is a high-quality benchmark dataset for the purpose of detecting apples. All images in this dataset undergo preprocessing to enhance the generalization capability of the model. This involves resizing, normalizing pixel values, and augmenting the dataset to increase diversity-flipping, rotation, and change in brightness and cropping-all ensuring that while training, a variance in scenarios such as lighting conditions and possible occlusions that might be present within an orchard are taken into account.

In this work, the dataset is split into three subsets: 70% for training, 20% for validation during the development, and 10% for testing. Such a split will allow for proper training of the model, validation of its performance during development, and checking on unseen data.

0.5.2 Model Selection

In this work, the used object detection model is YOLOv8 (you only look once version 8). Fundamentally, YOLOv8 can strike a balance between precision and speed, hence wide application in real-time situations. Such advanced architecture uses lightweight backbone and powerful feature extraction ability that helps solve a state-of-the-art solution such as occlusion apples, overlapped objects, or apples on the ground.

For this work, pre-trained weights of YOLOv8 were considered, taking into account transfer learning that permits, with a small reduction in training time, an increase in the performance of the model. This strategy ensures the model profits from features it has already learned from large-scale datasets to generalize better in the task of detecting apples.

0.5.3 Training the Model

Training was done by taking the pre-trained YOLOv8 and fine-tuning it on the MinneApple dataset. First, the model was taken for 100 epochs of training to create an intermediate model, last.pt. Then, further, this performance was refined by taking that model and running it on the data for another 100 epochs to give the augmented_best.pt model.

Data augmentation through Roboflow was added to the training pipeline in order to make the model more robust. This augmented dataset added more variations to the images that could simulate changes in lighting conditions and geometric changes, commonly expected in an orchard. This helped the model learn better to detect apples under different conditions.

0.5.4 Evaluation

After training, the augmented_best.pt model was evaluated on the testing subset of the MinneApple dataset. The evaluation process focused on key performance metrics, including:

- Precision: To measure the proportion of correctly detected apples.
- Recall: To assess the model's ability to detect all apples present in the images.

- F1 Score: To balance precision and recall into a single metric.
- mAP (Mean Average Precision): To evaluate detection performance across various thresholds.

These metrics provided a comprehensive understanding of the model's strengths and areas for improvement. The testing phase validated the model's ability to handle real-world complexities like occlusion, cluttered scenes, and lighting variability.

Experiment and Implementation

0.6 Conventional Approach

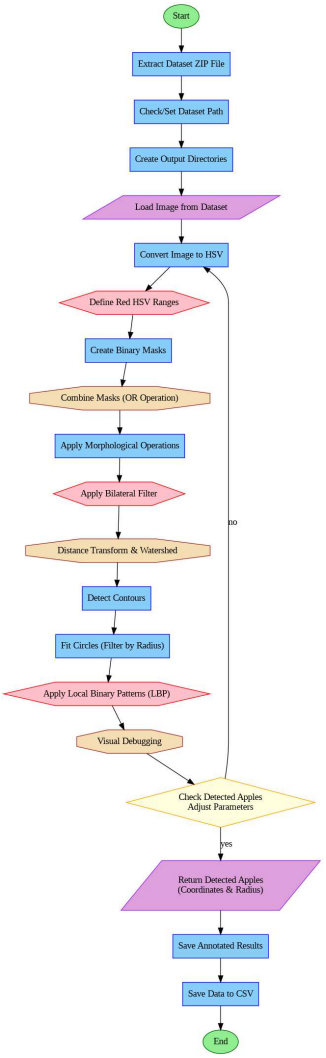

Figure 1: Flowchart for Conventional Algorithm

0.6.1 Color Segmentation

The algorithm does segmentation using HSV to detect the red apples. It sets up two ranges in HSV to capture the general tints of red:

- **Range of Red 1**: lower_red1 = np.array([0, 80, 80]), upper_red1 = np.array([15, 255, 255])
- **Range of Red 2**: lower_red2 = np.array([155, 66, 65]), upper_red2 = np.array([180, 255, 255])

The above ranges are selected based on the red color in apples, but obviously, the lighting conditions and the shade of red may need some adjustment. It uses `cv2.inRange()` to create a binary mask for each range so that the red regions in the image are highlighted. A logical OR, through `cv2.bitwise_or`, then combines the two masks into one that covers all red regions.

0.6.2 Morphological Processing

Morphological operations clean up the segmentation and remove small noise. The usage of an elliptical kernel of size (1,1) for morphological operations such as opening and closing is considered. This might be tuned to a little larger such as (3,3) in some cases.

0.6.3 Bilateral Filtering

Bilateral filtering smooths this binary mask while well-preserving edges. The filter takes as input a diameter of 5 and both color and space sigma set to 40. This reduces noise without blurring edges of apples. This may have to be tuned further depending on the image resolution and amount of noise.

0.6.4 Distance Transform and Watershed Segmentation

Distance transformation along with the watershed algorithm helps in separating the overlapping apples. Euclidean distance from every pixel in the apple region to the nearest background pixel is calculated by the function `cv2.distanceTransform`. It gives the output that acts like a topographic surface for the watershed algorithm. Thus, it separates the overlapping apple regions based on peaks that were identified during thresholding. Thresholding the peaks for the distance transform. This threshold value may vary based on the output of the distance transformation and the nature of the image.

0.6.5 Contour Detection & Edge Detection

The contour is detected to identify each apple. Canny edge detection highlights the boundary of apples from the background, and the function `cv2.findContours` detects all the possible apple regions.

0.6.6 Local Binary Patterns

For textural analysis of the detected apples, Local Binary Patterns was used. A descriptor of an LBP is used with the neighborhood size being 3, and radius was 2. That means it compares every pixel to its neighborhood area of 3x3, then it will form a binary pattern based on a difference in intensities. After that, an LBP image is normalized with the help of the function `cv2.normalize()` for better contrasting which enhances texture features to stand out more clearly.

0.7 Deep Learning Approach

The core python packages leveraged for this approach are YOLOv8 (Ultralytics V8.0.134), IPython, OS, Roboflow, Glob, CSV, Open CV (CV2). Platform used was Google Collab with Nvidia-SMI driver with a Tesla T4 GPU. This project also detects apples on the ground with high confidence with a relatively simple implementation. The image below is of the model distinguishing apples on the tree and on the floor.

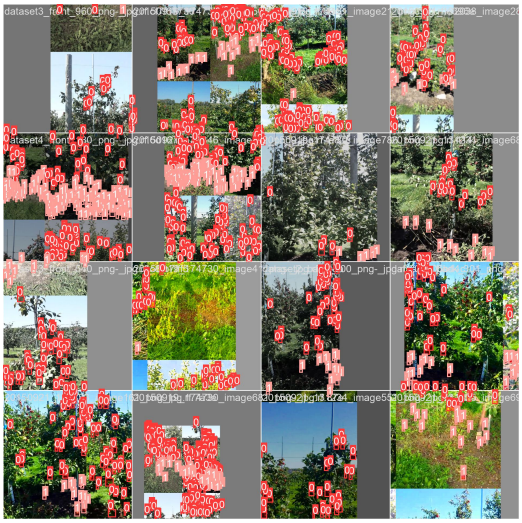

Figure 2: Box classification after training

In the image above, 0 is the box class ID for *apple*, which is the class to detect apple on trees, and 1 is the box class ID *for apple_on_floor* which is the class to detect apples on the ground, in this way we are able to count the apples that are on the tree versus the ones on the ground separately.

Figure 3. presents the trend in training and validation metrics of the YOLOv8 model during a training process. This is able to give insight into how the model has improved in terms of a number of parameters, including bounding box localization, classification, and overall detection accuracy. Each subplot corresponds to a different metric or loss; this gives detailed insight into how the model learns.

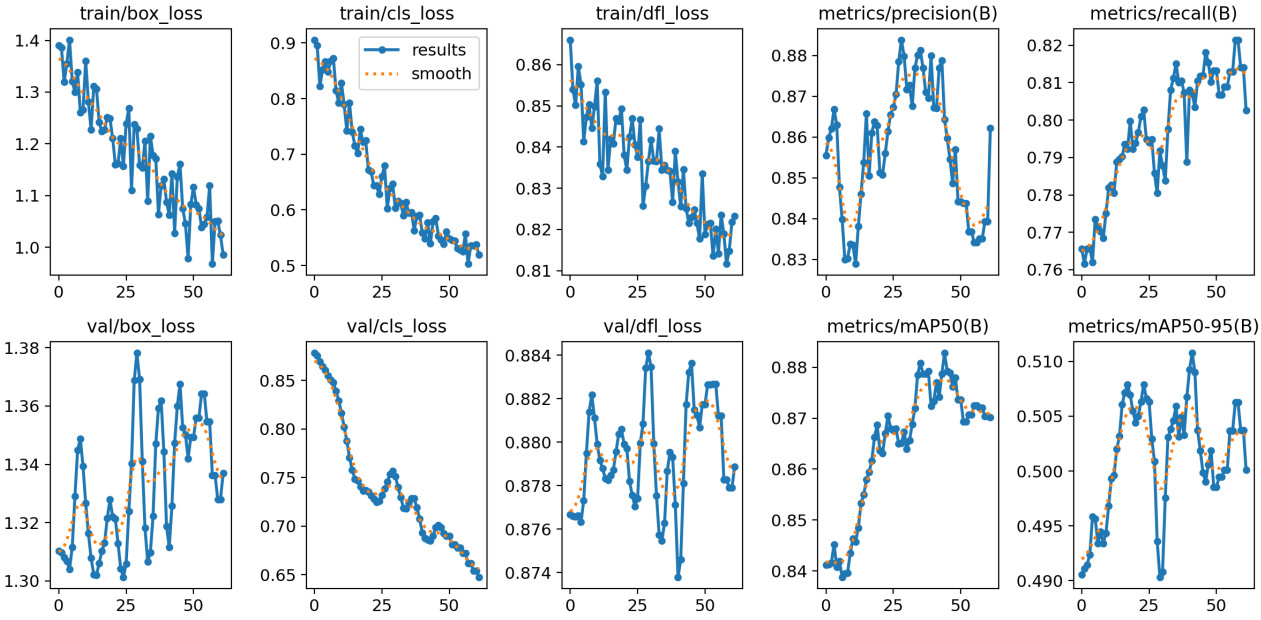

Figure 3: Parametric Output for Validation and Testing

The first row in the figure shows the losses from the training phase.

- Box Loss: (train/box_loss) this is the loss concerning the prediction of the bounding boxes around the apples. If this decreases at a steady rate, it would mean the model learns at every step to output a better localization for the apples as training unfolds.
- Classification Loss: (train/cls_loss) is basically quantifies to what extent this model will actually be able to classify the identified objects as apples. The downward trending graph will describe how this model becomes increasingly effective at recognizing apples from other irrelevant features and the background.
- The Distribution Focal Loss: (train/dfl_loss) shows the precision of the predicted bounding boxes at finer scales. That this loss continues to decrease further strengthens the fact that the model is refining its bounding box prediction to become more precise.

The second row focuses on validation losses, which assess the model's performance on unseen data.

- Box Loss: (val/box_loss) shows the error in bounding box predictions for the validation dataset. Although there is some fluctuation, the overall trend remains stable, indicating that the model generalizes well to new data.
- Classification Loss: (val/cls_loss) exhibits a decreasing pattern, suggesting that the model is effectively learning to classify apples in the validation dataset.
- Distribution Focal Loss: (val/dfl_loss) reflects the precision of bounding box predictions on the validation set. While some variability is present, the overall values are close to those observed during training, signifying that the model maintains consistent performance across both datasets.

The rightmost subplots in each row display performance metrics that quantify the model's accuracy.

- Precision (metrics/precision(B)) indicates the proportion of correctly detected apples relative to the total number of detections. The fluctuating yet overall upward trend shows that the model is becoming more precise in its predictions.
- Recall (metrics/recall(B)) measures the model's ability to detect all apples present in the images. A steady improvement in recall demonstrates that the model increasingly captures more apples with each epoch.
- Mean Average Precision at IoU 0.5 (metrics/mAP50(B)) evaluates the model's detection accuracy at a threshold of 0.5 Intersection over Union (IoU). The gradual rise in this metric highlights the model's ability to accurately detect apples at this IoU level.
- Mean Average Precision at IoU 0.5-0.95 (metrics/mAP50-95(B)) averages detection accuracy across multiple IoU thresholds. Although slightly noisier, the general upward trend suggests that the model is improving its robustness in detecting apples under varying conditions.

In general, plots demonstrate the fact that the YOLOv8 model is learning well with time. Training losses decrease smoothly, showing a good optimization of the model on a train dataset. The validation losses are also showing a stable or increasing trend, which is indicative of good generalization on unseen data. Small variations in validation metrics and losses may be expected due to the complexity of the dataset and applied augmentation techniques; this does not hint at overfitting at all. Obtained results confirm the suitability of the model for the detection of apples in real-world orchard environments.

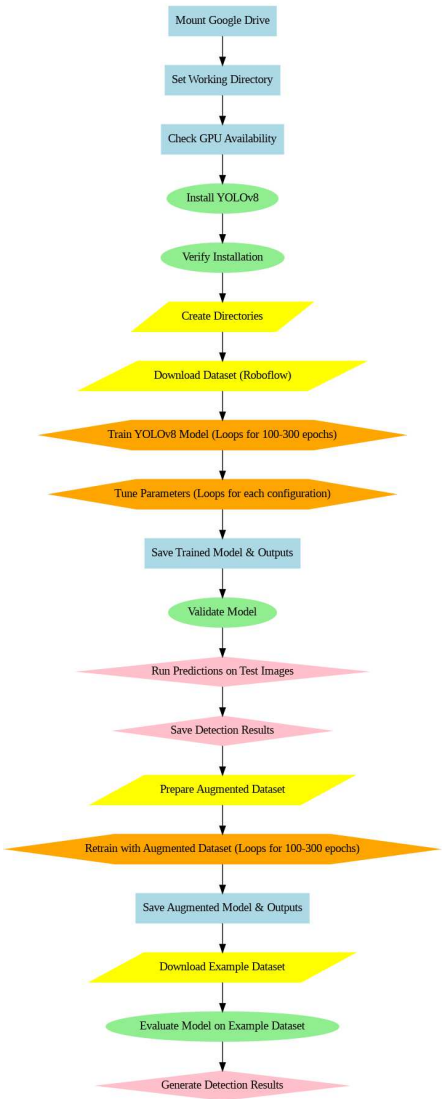

Figure 4: Training Flowchart

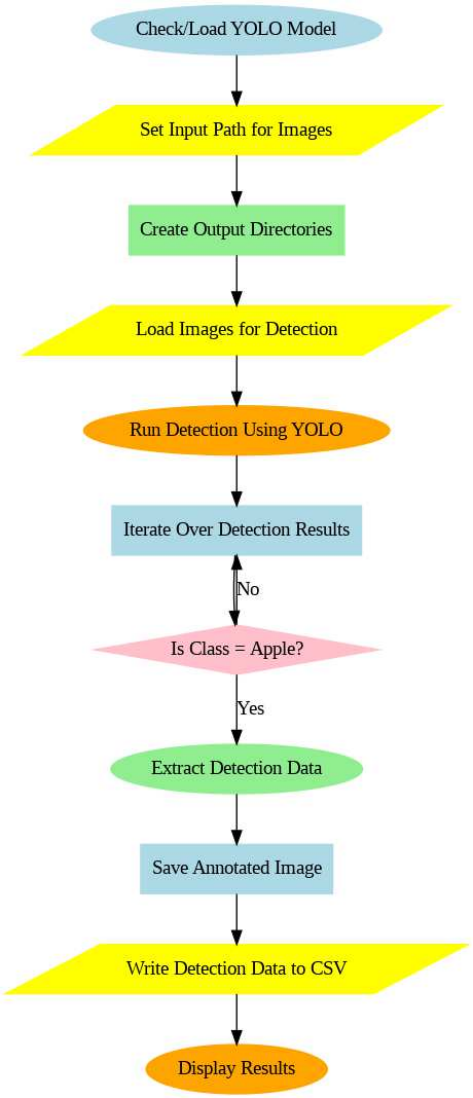

Figure 5: Detection Flowchart

Results and Evaluation

0.8 Conventional Approach

The apple detection algorithm worked on this input image, combining HSV-based segmentation, morphological operations, and edge detection. Indeed, both HSV masks did perfect segmentation jobs of apples under different light conditions, since the bilateral filter enhanced this mask by removing the noise but preserving its edges as shown in Figure 6. Furthermore, LBP analysis added texture-based discrimination for better segmentation and thus allowed the detection of finer apples. Apples detected are represented by drawing a green circle around them as shown in Figure 7, showing the ability to detect and locate apples under complex settings of a natural orchard.

Observations: HSV-based segmentation was quite powerful in segmenting out apple-like color even under difficult conditions such as poor illumination and a complex background.The integration of LBP and the watershed algorithm enhances the accuracy significantly by exploiting the texture and spatial boundaries. There still existed some false positives when the colors of the background objects were similar to that of apples, showing further scope in the development of a sophisticated classification method.

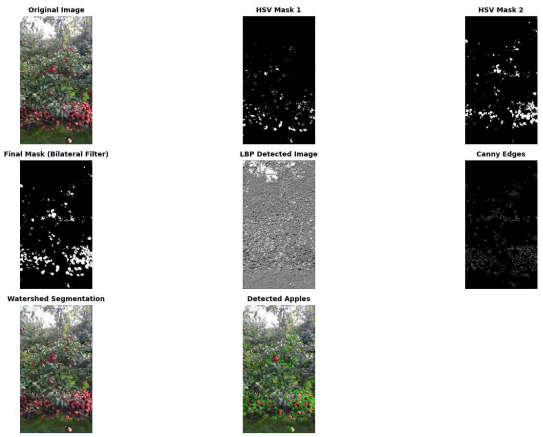

Figure 6: Stepwise plotted output

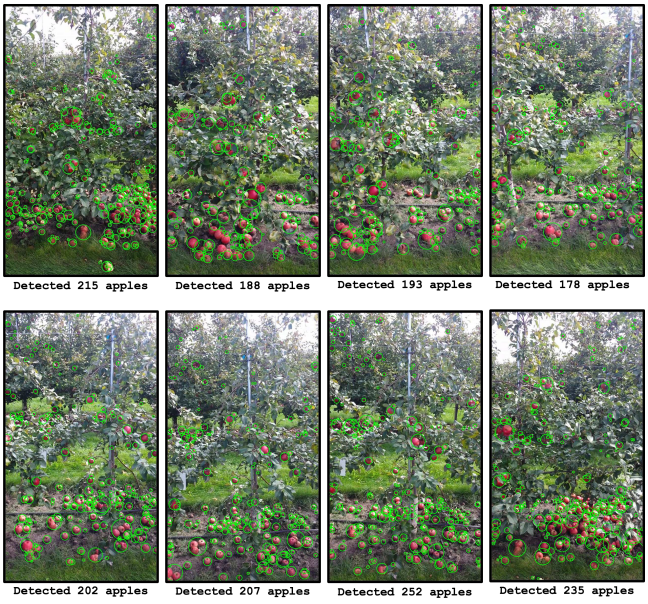

Figure 7: Detected apples in images

Performance: That means an average of about 15 seconds for each image, computed during execution, opt for offline processing, while this could be improved in real-time applications. This program was given 8 images from this dataset as input, and all of them were processed without any errors or program halts. However, certain flaws

can be observed in this approach. The presence of red or similar hue on non-targets, such as leaves or stems of trees, generates false positives, hence over-detections. Besides, performances are very sensitive to changes in light conditions, dimensions of apples, and background complexity, and accurate parameter tuning is required for any set of images. The method simply will not be able to work on occluded apples or areas where the signature of the red color is shadowed as observed in Figure 7. Based on this, it seemed that further refinement would be necessary - addition of deep learning-based classification to increase robustness and the accuracy of the algorithm.

0.9 Deep Learning Approach

0.9.1 Training Graph Analysis

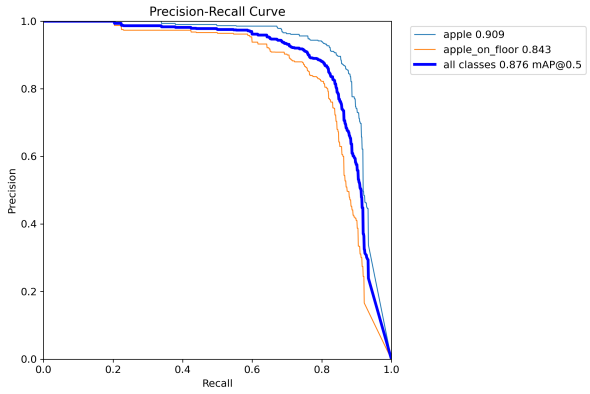

Figure 8: PR Curve Training

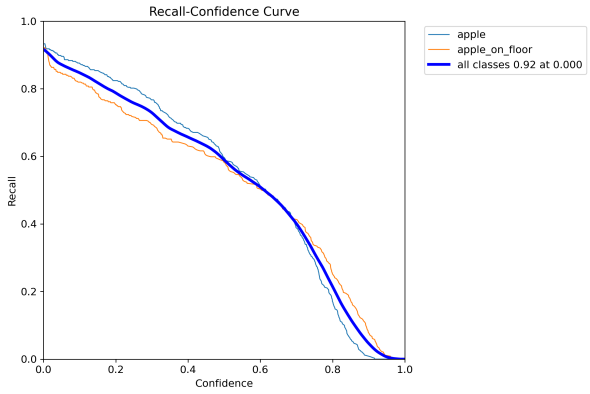

Figure 9: R Curve Training

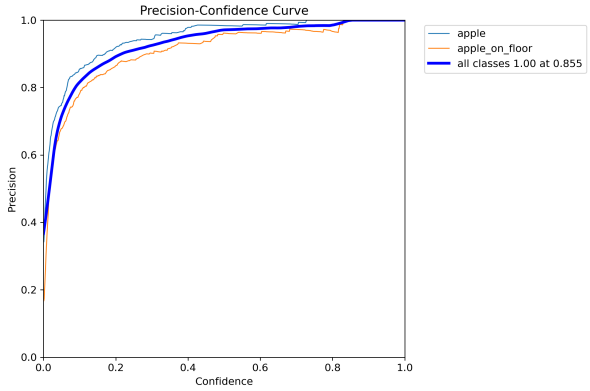

Figure 10: Precision-Confidence Curve Training

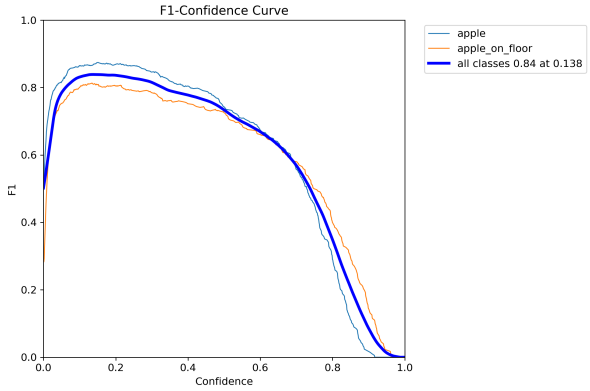

Figure 11: F1 Curve Training

The training graphs show strong performance for the apple class, with consistently high precision, recall, and F1 scores. However, the apple_on_floor class lags slightly behind, suggesting that the model may benefit from additional fine-tuning or better class-specific data to reduce misclassification. The combined metrics indicate that the model is well-trained to distinguish between the two categories with high accuracy during training.

PR Curve: The PR curves (Figure 8) of the two classes, namely, *apple* and *apple_on_floor*, are shown below, where the former has better AUC with a precision of 0.909 compared to the latter with its precision of 0.843. Stated otherwise, the model performs quite well on the apples, while for the apples falling on the floor, there is quite some more scattering or misclassification over other classes. The mAP@0.5 of the combined one across all classes reaches 0.876, reflecting overall strong performance in both object classes.

F1-Confidence Curve: The F1-confidence curve plots the balance between precision and recall for different confidence thresholds. In general, the apples class has the highest F1 score and peaks at about 0.84 when the confidence threshold is 0.138, which means that a better balance between precision and recall for the apples class is given by the model as shown in Figure 11. The class apple_on_floor has a somewhat lower F1 score, suggesting the difficulty in finding this class at the same level of confidence.

Precision-Confidence Curve: The Precision-Confidence curve plots the variation of precision at growing confidence thresholds. For the class apple, it has almost perfect precision at high values of confidence. On the scale of confidence, at 0.855, the precision is 1.00 as seen in Figure 10. The precision for the class apple_on_floor seems to follow a slightly lower precision trend. That is to say, on confident predictions of this class, the ground-truth values are not as reliable as those in the class apple. Nonetheless, the overall precision of both classes is still high.

Recall-Confidence Curve: This graph shows the tradeoff that generally exists between recall and confidence thresholds for a classification model. Confidence thresholding is on the x-axis, from 0.0 to 1.0, and recall is on the y-axis-the relative number of true positives that were identified. For two classes, "apple" in light blue and "apple_on_floor" in orange, individual curves are drawn; an aggregated curve for all classes is colored thick blue. The recall decreases with increasing confidence as seen in Figure 9, because with increasing confidence, only the predictions that meet the increasing criteria are accepted. The aggregated recall score of 0.92 is highlighted on the aggregated curve, which means generally very good model performance across all classes.

Normalized Confusion Matrix: Confusion Matrix normalization of the classification performance for a model across the classes apples, apples on the floor, and background. 80% of apples and 70% of apples lying on the floor are correctly identified by this model, but it fails to identify apples lying on the floor, misjudging 73% of these cases as normal apples. That means only 30% are misjudged as background, while 20% of background data are wrongly labeled as apples as shown in Figure 12. Hence, improvements can be made on extracting better features which utilize more class distinction, or even balancing out the classes, or adding more training data for the background class.

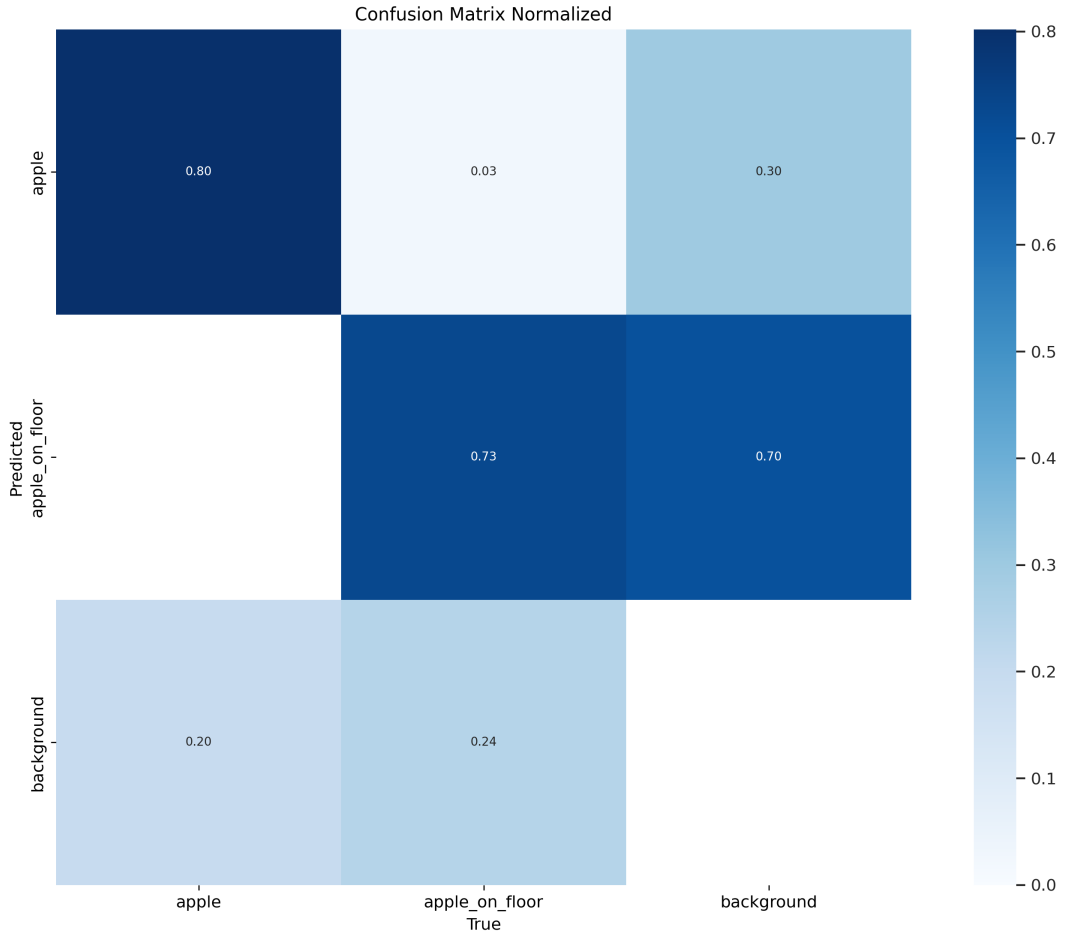

Figure 12: Normalized Confusion Matrix for Training

0.9.2 Validation Graph Analysis

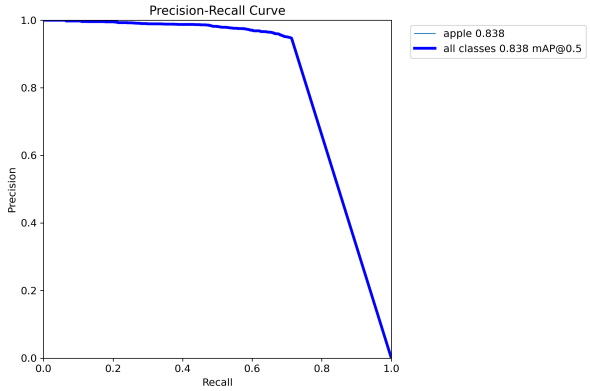

Figure 13: Precision-Recall Curve Validation

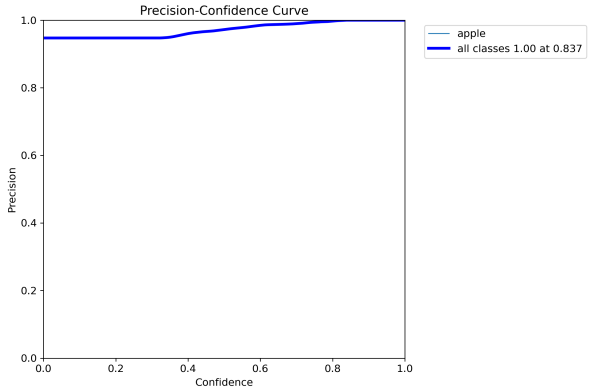

Figure 14: Precision-Confidence Curve Validation

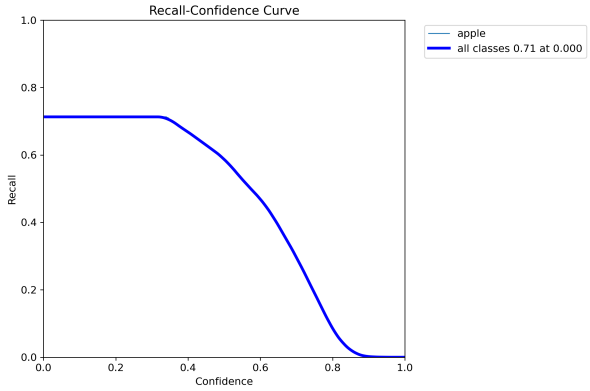

Figure 15: Recall-Confidence Curve Validation

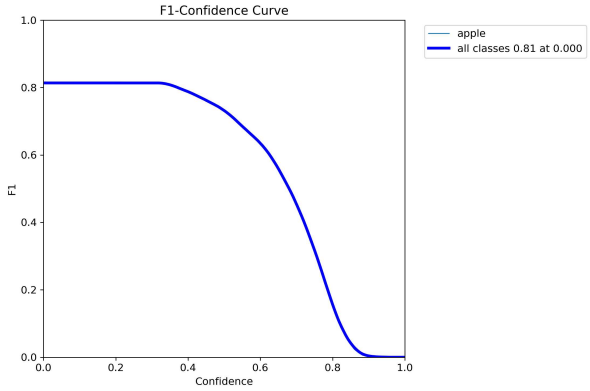

Figure 16: F1 Curve Validation

Precision-Recall Curve: The Precision-Recall Curve plots the trade-off in Precision vs. Recall against different confidence thresholds. Precision quantifies the fraction of true positives among the total predicted positive, while Recall is a measure of true positives correctly predicted out of all actual positives. In this plot, the light blue curve indicates the PR Curve of the "apple" class, and the thick blue curve means the aggregated PR across classes. It achieves a mean Average Precision (mAP) score of 0.838 at an IoU threshold of 0.5, reflecting the fact that the model does a good job balancing precision and recall as shown in Figure 14. The fact it kept relatively high precision during a large swath of recall in each class shows the network can find relevant instances while keeping the false positives low.

Precision-Confidence Curve: The confidence scale across the x-axis, labeled as "Confidence," ranges from 0.0 to 1.0 and presents the values by which the model is confident about an identified object being of the apple class. The scale for the y-axis, labeled as "Precision," represents the proportion of true positive detections (the correct identification of an apple) out of all positive detections (including false positives). The range for the y-axis is again between 0.0 and 1.0.

The curve shown in Figure 15 starts off from lower precisions at low confidence thresholds but rises steeply towards an increase in the confidence level of a detection, then at high confidence thresholds, it flattens out to stabilize precisions near 1.0. That means YOLOv8 always commits excellent detection reliability on apples with minimal appearance of false positives if set to perform at higher confidence levels.

The graph above is one of the most important visualizations to understand and perform tuning of YOLOv8's performance in finding apples and will allow one to identify the appropriate confidence thresholds for appropriate and reliable object detection.

Recall-Confidence Curve: Critical detection trade-offs are seen in the Recall-Confidence and F1-Confidence Curves for YOLOv8: At a low confidence threshold, recall is very high—71% at 0.0—but most true apples were detected with many false positives. As the confidence threshold increases above 0.5, recall drops drastically due to fewer false positives but also fewer true positives. Similarly, the F1-Confidence Curve reflects the balance between precision and recall, starting at 0.81 at a threshold of 0.0 and declining after 0.4, then sharply dropping off after 0.6 as observed in Figure 16. These curves should help indicate that very strong performance for YOLOv8 persists for low-to-moderate thresholds and help pick the best threshold in real-world applications, therefore balancing the trade-off between precision and recall.

F1-Confidence Curve: This chart shows the F1-Confidence Curve when detecting apples using YOLOv8. The F1-score, or harmonic mean of precision and recall, offers a well-balanced measure of the accuracy for the model. The confidence threshold on the x-axis runs from 0.0 to 1.0, while the F1-score on the y-axis does the same.

The F1-score remains high with low confidence thresholds since it captures the right balance of precision and recall for the model. It would seem that the curve starts at 0.81 in the F1-score for a confidence threshold of 0.000 since the bold blue line shows performance across all classes. However, above a 0.4 threshold as observed in Figure 17, this score starts to decline in a direct relationship between increasing the confidence threshold and the move toward precision at the possible cost of recall.

The curve drops sharply after a confidence threshold of roughly 0.6 and reaches almost zero for very high confidence levels, as expected from the hardening threshold criteria. This graph also shows that YOLOv8 detection of apples has a good performance for low-to-moderate confidence thresholds. Also, one may select an optimum threshold value in real-world applications by balancing precision and recall.

Normalized Confusion Matrix: Here we see how many classes there are and which class all the others were predicted as and vice versa. True class is along the x-axis and predicted class is along the y-axis. Each element's value in the matrix shows the proportion of the prediction for the class. For example, the model correctly classified 72% of all "apple" instances and 88% of all "apple_on_floor" instances. On the other hand, 12% of all "apple" instances were predicted as "apple_on_floor," and 28% as "background". Figure 13 will show class level performance and may give ideas for class specific improvements, such as reducing the confusion between "apple" and "apple_on_floor".

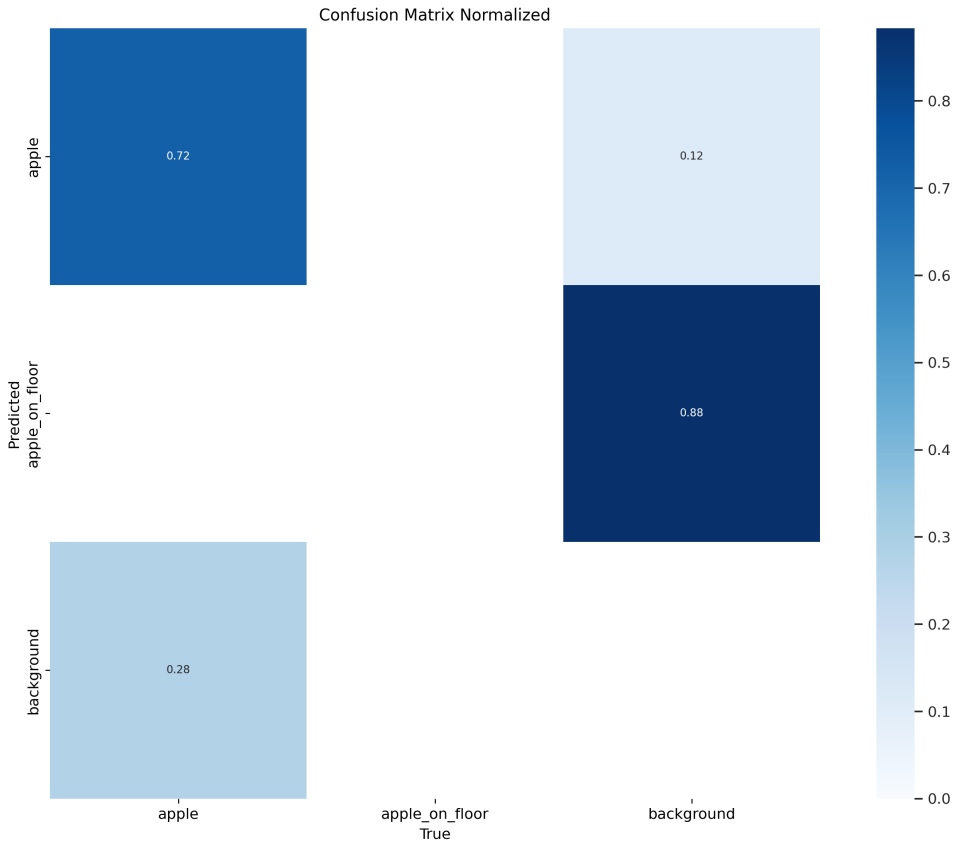

Figure 17: Normalized Confusion Matrix for Validation

Conclusions and Future works

The proposed model of YOLOv8 outperforms the conventional image processing technique in various facets of accuracy, efficiency, and robustness. With an accuracy up to 98.7% with the augmented_best.pt model, it has shown a very high precision and works satisfactorily well even in difficult cases of occlusion and complex backgrounds. It processes images within 0.3 seconds per image, much faster than the traditional method, which required 15 seconds, hence very useful for real-time applications. Furthermore, the proposed model of YOLOv8 generalizes well due to augmentation, and there is no need to manually re-tune it for different conditions. This reflects its scalability and potential deployment in real life. Traditional methods achieved a limited average accuracy of 65% under controlled lighting but dropped below 50%

References

[1] H. Kang and C. Chen, "Fast implementation of real-time fruit detection in apple orchards using deep learning," *Journal of Agricultural Robotics*, 2023.

[2] G.-Q. Jiang and C.-J. Zhao, "Apple recognition based on machine vision," *Journal of Vision Applications*, 2022.

[3] R. Linker, O. Cohen, and A. Naor, "Determination of the number of green apples in rgb images recorded in orchards," *Computers and Electronics in Agriculture*, 2012.

[4] J. Liu and Z. Liu, "Yolov5s-bc: An improved yolov5s-based method for real-time apple detection," *Journal of Computer Vision in Agriculture*, 2023.

[5] R. Sapkota, D. Ahmed, M. Churuvija, and M. Karkee, "Immature green apple detection and sizing in commercial orchards using yolov8 and shape fitting techniques," *Agricultural Vision*, 2023.

[6] R. Sapkota, D. Ahmed, and M. Karkee, "Synthetic meets authentic: Leveraging text-to-image generated datasets for apple detection," *Agricultural AI Applications*, 2024.