HIGH SCHOOL EDITION
Journal of Student Research

Volume 14 Issue 1 (2025)

Apple Recognition Method in Orchards Based on SAM and YOLOv8

Sunny Lu[1] and Zhiquan Jiao[#]

[1]Point Grey Secondary School, Vancouver, British Columbia
[#]Advisor

ABSTRACT

The aim of this research is to improve automatic harvesting of orchard apples through an efficient detection method. By applying TensorRT, the YOLOv8 model will run much more efficiently while optimising computational resources. In particular, we believe that the depth and complexity of various YOLOv8 versions of the model will play a key role in improving the detection performance. Therefore, in this study, we tested several versions of YOLOv8 algorithmic recognition models, such as YOLOv8s, YOLOv8n, YOLOv8l, YOLOv8m, etc., and used a variety of model annotation methods including traditional manual annotation, unsupervised annotation, and semiautomatic annotation tools based on the large-scale SAM model SAMsaa. In addition, we tested the effectiveness of automatic apple detection in orchards with and without hardware acceleration. In order to test the above hypotheses, we conducted several experiments and showed that the overall detection performance of the YOLOv8m model was significantly improved in the experimental setting where the dataset was labelled using the SAMsaa tool and optimised using TensorRT. In addition, the overall detection performance of the YOLOv8m model was even more significantly improved in the experiments where the TensorRT-optimised dataset was labelled using the SAMsaa tool on the Jetson Xavier computing platform. The detection mAP50 improved by 33% and 32.7%, respectively, and the average detection accuracy for apple detection reached 90.41%. These results validate the effectiveness and superiority of our method.

Introduction

This study focuses on apple picking automation, specifically image annotation and target detection acceleration on edge computing devices for picking robots. Traditional apple picking relies on a lot of manual labor, which is laborious, expensive, and prone to damage to the fruit. With advances in technology, mechanized and automated harvesting research can increase efficiency, reduce reliance on labor, lower costs, and enable sustainable agriculture. Precision agriculture techniques optimize the picking process by monitoring orchards to improve yield and quality. However, the level of automation still needs to be improved, such as efficient harvesting, handling of irregular fruits, and stable operation in different environments. An in-depth study of apple harvesting technology is essential to improve industrial competitiveness and modernize agriculture.

Picking techniques based on target detection algorithms include histogram of oriented gradients HOG (Dalal et al. 2005) and convolutional neural networks (CNN). Although HOG performs well, its fixed detection window size limits performance. Convolutional neural networks such as AlexNet (Krizhevsky et al. 2012), VGG (Simonyan et al. 2014), GoogleNet (Szegedy et al. 2015), and ResNet (Szegedy et al. 2017) have improved object detection performance. Early detection algorithms used selective search algorithms to generate candidate boxes. With the improvement of CNNs, orchard apple detection methods include Region-based Convolutional Neural Networks (RCNN) (Girshick et al. 2014) and Faster R-CNN (Ren et al. 2016). In 2016, Redmon et al. proposed you only look once (YOLOv1) (Redmon et al. 2016), which improves the detection speed by dividing the grid for bounding box regression and category prediction, and using convolutional neural network feature extraction. Subsequently, YOLOv2 (Ren et al. 2017) and YOLOv3 (Ren et al. 2018) improved detection accuracy and speed. Recently, Glenn-Kocher et al.

introduced YOLOv5, which improved the detection performance by introducing Focus and Cross Stage Partial Network (CSP) modules as well as the Feature Pyramid Network (FPN) + Path Aggregation Network (PAN) structure. However, when using the above methods for orchard apple identification, the overall identification accuracy and processing efficiency of these algorithmic models were found to have potential for improvement.

The main contributions of this research include: proposing a system optimisation method and validating it on a small edge device, using an interactive system based on the SAM macromodel to efficiently annotate the data, selecting YOLOv8m as the best detection model through comparative experiments, evaluating and selecting the Jetson Xavier NX as the edge device, and accelerating the inference using the TensorRT optimiser. We hypothesize that orchard apples can be detected more efficiently and quickly by using a semi-automatic annotation tool based on large SAM models (SAMsaa) for dataset annotation and deploying the TensorRT-optimized YOLOv8 model on a Jetson Xavier NX edge computing device. To test this hypothesis, we conducted the following experiments: comparing data from different annotation methods, deploying the Jetson edge device for testing, and testing the performance of different algorithm variants.

Methods

Experiment

Experiment Step 1. Data Collection Using Industrial Cameras

In order to collect image datasets of actual apple picking orchard scenes in this study,By deploying the RealSense above D405 camera in everyday Apple detection environments, real-time information about Apple is collected by separating frame-by-frame data from videos and saving the collected video data.

Experiment Step 2. Dataset Processing

The SAMsaa semi-automatic annotation tool was deployed on the Ubuntu system to annotate the collected datasets.

Experiment Step 3. YOLOv8 Model Training

For the comparison work of apple detection models, several variants of the YOLOv8 target detection algorithm were selected for this experimentThe dataset of 10,000 images obtained by semi-automatic labeling tool is divided into 90% and 10% as training and validation sets respectively. The training set is used for model learning, while the validation set is used to evaluate model performance and prevent overfitting. 90% and 10% is a common division that provides a reasonable validation sample size while ensuring sufficient training data.

Experiment Step 4. Deployment And Testing of Jetson Edge Devices

Considering factors such as market performance and price of edge devices, this paper uses Jetson Xavier NX devices and TensorRT technology for inference acceleration. Including: firstly, train the model on the PC device according to the set parameters, and get a model file in pt format after completion. Secondly, transfer the model file to Jetson, and convert the pt model to was format model file under PyTorch environment. Finally, the WTS file is converted into an engine format model file for Jetson device detection.

Method

This paper explores the automation of the apple picking process with a focus on image data annotation and target detection acceleration for edge computing devices that can be deployed on picking robots. The paper proposes a system optimization method and provides preliminary validation of the deployment of the algorithm on a small edge computing device. In terms of data processing, the thesis combines an interactive dataset labeling system based on the

SAM large model to efficiently and accurately complete the annotation of image data. In terms of the selection of object detection model, the paper conducts multi-version comparison experiments of YOLOv8 model, and selects the most suitable end-to-end detection model for this research -- YOLOv8m model, which enhances the apple detection method in this paper. In terms of deployment in real-world environments, this paper evaluates the deployment results of multiple models on edge devices, and selects the Jetson Xavier NX edge device after comprehensively considering various factors. It uses the TensorRT high-performance deep learning inference optimizer to accelerate the inference process, significantly improving the overall method's speed and accuracy.

SAM Network Structure

The SAM (Segment Anything Model) network model is derived from the Segment Anything (SA) project proposed by Meta in 2023. Researchers are dedicated to constructing a comprehensive model for image segmentation.The workflow of the SAM network model is divided into three main steps: Task, Model, and data (Figure 1).

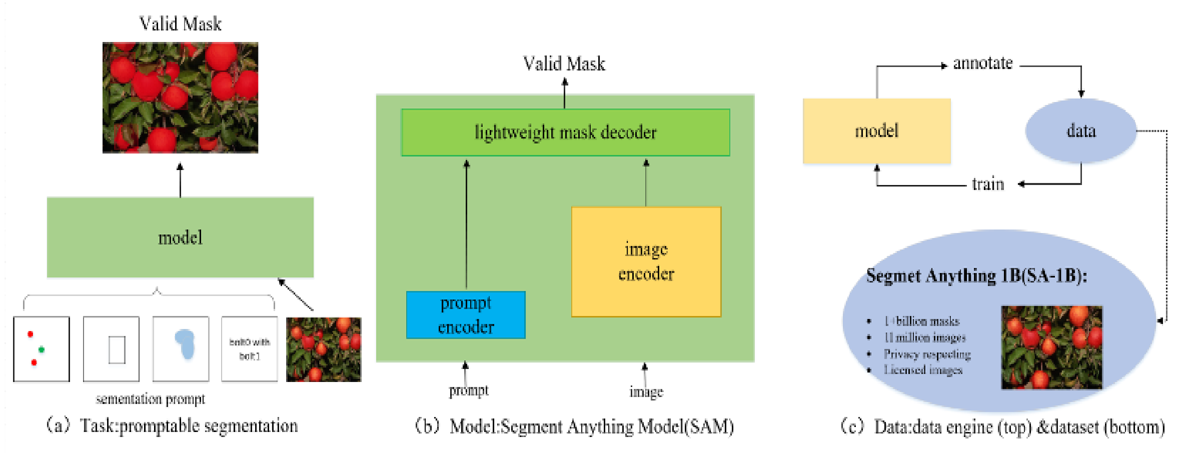

Figure 1. SAM Network Model.

① Task

SAM is trained for segmentable tasks with prompts in the task module,using a huge dataset, where the number of dataset images is nearly a million and contains more than a billion masks, This enables SAM to generate effective segmentation masks for any prompt.

② Model

In terms of model structure, Profiled at the model architecture level, the SAM large model skillfully incorporates three core components: an image encoder, responsible for parsing and extracting key information from the image; a cueing encoder, focused on encoding external cues or conditions into a format understandable to the model; and a mask decoder, responsible for generating or recovering the masked portion of the image based on the output of the first two, thus accomplishing complex image processing or generation tasks ,as shown below (Figure 2).

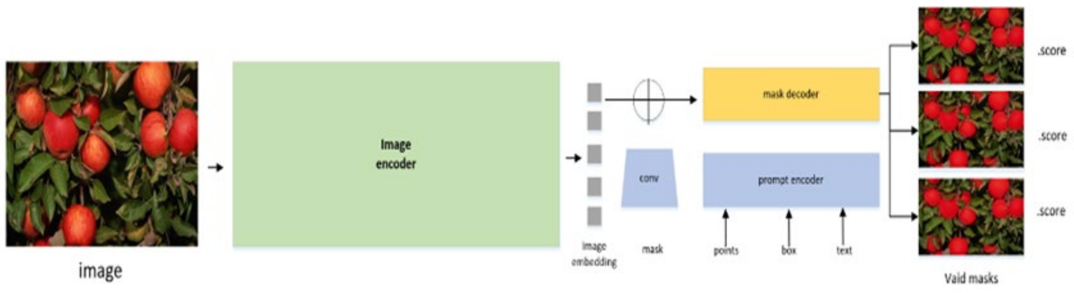

Figure 2. General structure of the SAM model**.**

Image Encoder: Researchers use ViT pre-trained with MAE, a scalable and robust pretraining method for processing high-resolution input images. Each image goes through the image encoder once before being passed to the prompt Model.

Prompt Encoder: In designing the cueing encoder, the researchers explored two different properties of cueing inputs, one of which is discrete and sparse cueing types that provide guidance to the model in a discontinuous or sparse form, and the other is continuous and dense cueing, which is an approach that conveys information or directs the focus of attention of the model by directly acting on a specific portion of the input data in a densely covered form. With these two different types of cues, the researchers aim to optimize and enhance the performance and adaptability of the encoder.

Mask Decoder: The mask decoder generates a segmentation mask of the same size as the input image by fusing the features of the image encoder and the cue encoder, ensuring an accurate distinction between the target object and the background.

The overall structure is shown in Figure 3 below.

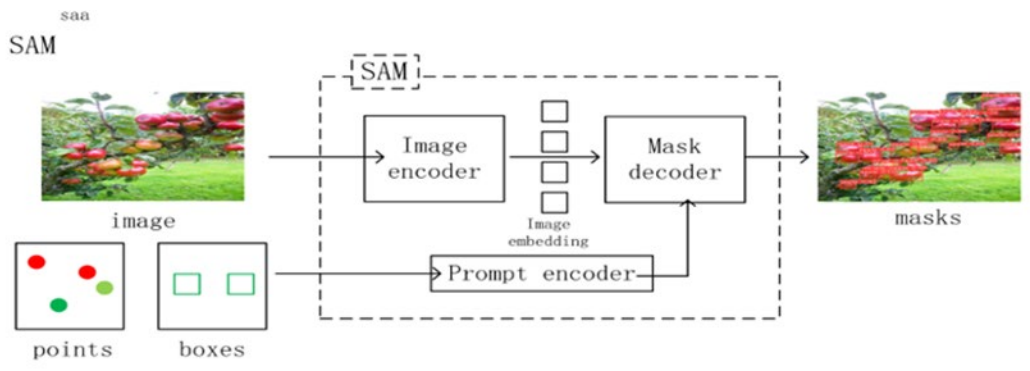

Figure 3. Structure of Image Encoder, Prompt Encoder and Mask Decoder.

③Data Engine

Since the original splitting masks are limited in variety, the SAM network model proposes a data engine result.According to the degree of automation of data annotation, the data engine can be divided into: firstly, the manual annotation stage assisted by the model, followed by the semi-automated annotation process, and finally, a fully automated annotation stage, which gradually reduces the dependence on manual intervention while gradually improving the annotation efficiency and model autonomy.

(1) Model-Assisted Manual Annotation

Before this phase begins, the SAM model is trained using standard public image segmentation datasets. Then, SAM uses this trained Model to predict image masks in the SA-1B dataset. Subsequently, a group of professional annotators refines the predicted masks. These annotators have the freedom to add labels to the masks and must mark them according to the importance of the objects.

At the same time, as the Model improves, the average annotation time for each mask is reduced from 34 seconds to 14 seconds. This improvement makes the mask annotation speed of SAM 6.5 times faster than that of COCO but still two times slower than 2D box annotation.

(2) Semi-Automatic Phase

In the data engine of the SAM model, the semi-automatic annotation phase starts with the generation of the initial segmentation mask by the pre-trained model, followed by the necessary adjustments by the human annotator. This stage significantly reduces the manual workload while maintaining high annotation accuracy, laying the foundation for the eventual realization of fully automatic annotation.

(3) Fully Automatic Annotation Phase

The fully-automatic labeling stage relies entirely on the model to independently generate segmentation masks without human intervention. Through the high-quality data and continuous optimization in the first two phases, the model can efficiently and accurately complete the annotation task in this phase, which significantly improves the annotation efficiency and autonomy.

Semi-Automatic Annotation Tool Based on SAM Network and LabelStudio

LabelStudio is a versatile open-source data annotation tool that provides users a flexible platform for quickly creating, managing, and supervising annotation tasks. The design philosophy of LabelStudio is to abstract the complexity of data annotation tasks, thereby simplifying the entire data annotation process (Figure 4).

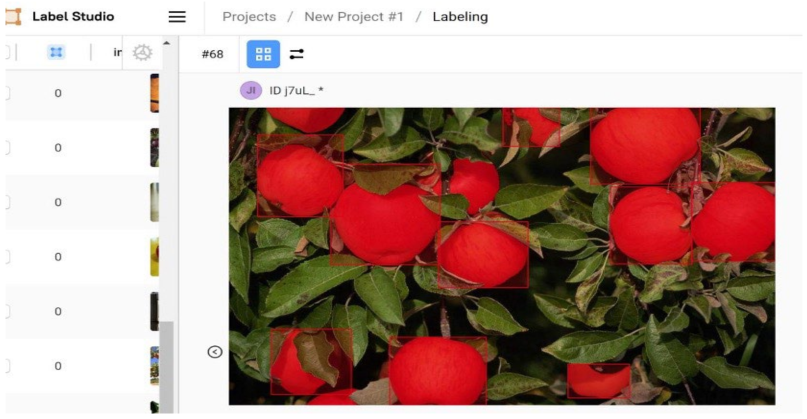

Figure 4. Label-Studio Tool Interface.

The SAM network-based semi-automatic annotation tool (SAMsaa, SAM Semi-automatic annotation tool) is divided into frontend and backend deployments.

In the backend part, it is necessary first to start the LabelStudio ML tool with label-studio-ml start sam and set the project to SAM. Then, it requires specifying a configuration file for LabelStudio ML. In this tool, the SAM model is applied to machine learning annotation, so it is necessary to set the corresponding training model's weights and parameters for loading the Model.

To generate the final output of the semi-automatic annotation tool, the Mask part and the corresponding bounding box need to be enabled. It is required to use out_mask=True and out_bbox=True. Since the SAM model requires significant computational power when running, the device's CUDA must be utilized. On the front end, The Label-Studio web service is initiated with a Label-Studio start (Figure 4).

The SAMsaa tool feeds raw images (such as the Apple image used in this paper) into a powerful image encoder of the SAM network model to compute image embeddings. This component extracts the image feature information mainly through Convolutional Neural Networks (CNN) or other deep learning frameworks such as ResNet and VGG.

After extracting features from the original apple images in the Image Encoder and obtaining encoded images, corresponding lower-dimensional vectors are generated. These vectors, known as image embeddings, are typically fixed-length numerical vectors rich in semantic information.

In addition to inputting the corresponding original Apple images into the Image Encoder module, the SAMsaa tool also needs to input prompts of box type or point type into the Prompt Encoder module to obtain the feature space of the prompt. Then, attention fuses these with the embeddings obtained from the Image Encoder in the same channel.

On the other hand, the Mask Decoder (mask decoder) is responsible for converting the features and information learned by the model into image segmentation masks.

Orchard Apple Recognition Based on the YOLOv8 Network

YOLO (You Only Look Once) is currently a very popular algorithm in the field of target detection, which simplifies the complex detection task into a straightforward regression process that simultaneously outputs the bounding box coordinates of the target object and its category through only a single forward computation process. The algorithm extracts image features through multiple layers of convolution and pooling operations. It outputs a three-dimensional tensor containing all grid cells, which includes information on bounding boxes and class probabilities. Non-maximum suppression algorithms are used to filter out redundant bounding boxes.

YOLO offers faster speed and excellent real-time performance compared to traditional object detection algorithms. Although there might be some limitations in dealing with densely packed and varied scale objects, it excels in detecting small objects and has significantly impacted the object detection field. The latest version, YOLOv8, integrates multiple SOAT (Self-Occlusion et al.) model technologies, offering more significant advantages in practical applications (Figure 5).

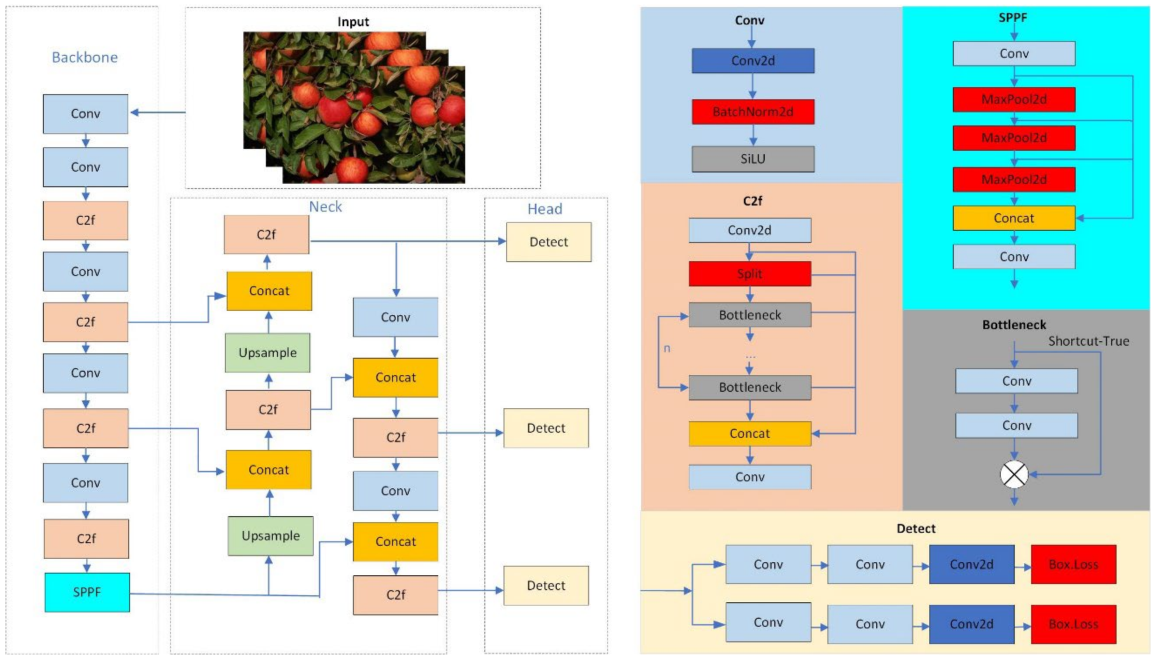

Figure 5. YOLOv8 Framework Structure Diagram.

The Backbone part is responsible for extracting critical and expressive feature information from the image, and a key change has been made in the latest YOLOV8 algorithm, which employs the lightweight C2f module, enhancing the ability to express features through a dense residual structure.

In the neck region of the target detection architecture, one of the main tasks is feature fusion. By cleverly utilizing the Path Aggregation Network (PANet) and the C2f module, this region is able to integrate feature maps from three different phases of the backbone network, which carry information at different scales. This fusion strategy effectively combines the detail-richness of shallow feature maps with the high-level semantic information of deeper feature maps, thus enhancing the overall feature representation.

As for the head part, it adopts an innovative decoupled head design, which decomposes the detection task into two independent but collaborative branches: the classification branch and the position prediction branch. This design aims to mitigate potential conflicts between classification and location tasks in traditional detection heads, allowing each branch to focus on its own task, thus improving overall detection performance.

In terms of loss calculation, YOLOv8 uses Varifocal Loss and Complete IoU Loss + Distribution Focal Loss. Based on the focal loss function, Varifocal loss better handles class imbalance and improves detection accuracy. CIOU Loss better handles the overlap between predicted and actual boxes. DFL Loss can better deal with class imbalance and background class issues.

Acceleration on Jetson Edge Devices

Deep learning techniques are being optimized day by day, the demand for real-time inference on mobile devices is also increasing. In the apple detection study of this paper, since the experimental method proposed needs to ensure the accuracy of apple picking, the real-time requirement of the method is self-evident. However, deep learning models typically use fairly large computational resources when running. These resources include, but are not limited to, high-performance processors, large amounts of memory space, and possibly specialized graphics processing units (GPUs) to accelerate the computational process. To solve this problem, this study uses NVIDIA's Jetson platform in combination with the TensorRT inference engine for real-time inference verification on edge devices, preparing for future deployment of Jetson devices on robots. The Jetson platform is equipped with high-performance GPUs, and TensorRT can optimize deep learning models to improve inference speed. This research will explore how to fully leverage the Jetson platform's and TensorRT's potential to accelerate edge devices in the experimental part.

Discussion

The aim of this study is to analyze the effects of various model variants. and methods on the detection effectiveness of orchard apples. In terms of modeling, we compared the effects of model depth and complexity on detection effectiveness in similar target detection models. We selected four model variants, YOLOv8s, YOLOv8n, YOLOv8l and YOLOv8m, for evaluation. It was hypothesized that the depth and complexity of the models would lead to more accurate results compared to the other test factors in the devices/models. Models with higher depth and complexity typically have stronger feature extraction capabilities and are able to capture more detailed image features, and YOLOv8m excels in this regard, balancing depth and complexity so that it can provide highly accurate detection results while maintaining high real-time performance. According to previous research, as depth and complexity increase, the effectiveness of deep learning models tends to increase accordingly, with more layers and parameters enabling the model to learn more complex patterns and improve detection accuracy.

In terms of dataset processing, we chose a variety of dataset labeling methods and assumed that the use of semi-automatic labeling tools could make the data labeling methods more efficient and faster. Comparison experiments of various labeling methods showed that the labeling efficiency based on the SAM model was greatly improved compared with manual labeling, and that the automatic labeling capability of the SAM model made large-scale dataset labeling more efficient and accurate. In contrast, manual annotation has the problems of time-consuming and high labor cost, while the application of SAM model can greatly reduce the burden of these problems, thus accelerating the progress of dataset compilation. The advantage of SAM model also lies in its ability to automatically deal with a variety of images, whether it is the view angle, lighting, occlusion and other issues, it can better cope with, so as to ensure the accuracy and consistency of the annotation. This makes the SAM model more widely applicable in practical applications, able to cope with a variety of complex annotation scenarios, and provide a reliable data base for subsequent deep learning model training.

In terms of hardware deployment, this study tested a variety of edge-end deployment methods, and we assumed that by using the Jetson Xavier NX edge device at the edge end and using the TensorRT acceleration method, the inference speed can be effectively improved. In the comparison experiment, initially, Without acceleration using TensorRT technology, YOLOv8 models tend to take a long time in the inference phase, while at the same time consuming significant GPU computational resources.However, after TensorRT optimization, the inference time is much shorter and the GPU resource usage is reduced. Compared to the unoptimized case, the optimized YOLOv8 model doubles the inference speed while maintaining accuracy.

However, it's crucial to consider the potential challenges and limitations of these methods in real-world applications. While the YOLOv8m model showed promising results, its increased computational requirements may limit its use in resource-constrained environments. The SAM-based labeling tool, although efficient, may struggle with highly occluded or unusually shaped apples, potentially introducing biases in the dataset. Additionally, while TensorRT optimization significantly improved inference speed, it may not be universally applicable to all edge devices or model architectures.

Future work should focus on addressing these limitations, perhaps by exploring model compression techniques for more efficient deployment, improving the robustness of semi-automatic labeling tools, and investigating alternative optimization methods for a wider range of devices. Furthermore, real-world testing in diverse orchard conditions is necessary to validate the model's performance under varying lighting, weather, and apple variety scenarios. Lastly, the economic feasibility of implementing these advanced technologies in agricultural settings should be carefully evaluated to ensure practical adoption.

Results

We annotated the dataset using the SAMsaa tool and the traditional tool manual annotation tool and we got the results as shown in Table 1 below.

Table 1. Comparative experiment of various annotation methods

Way	Tool	Time
Manual annotation	Lableimg	500 min
semi-automatic	SAMsaa	**235min**

We selected YOLOv8s, YOLOv8n, YOLOv8l and YOLOv8m models for training respectively. Each model variant was trained using the same dataset, the weights of the trained models were applied and validated, where the main focus was to show the different losses of the YOLOv8m model (convergence curves of YOLOv8m on the dataset obtained using the SAMsaa tool (Figure 6).

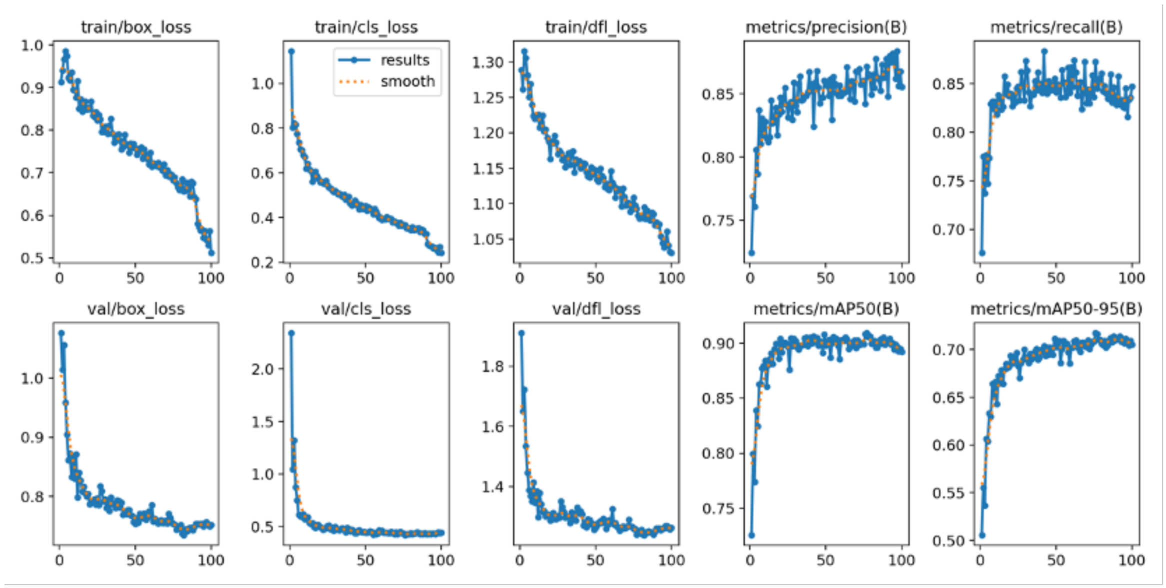

Figure 6. Convergence curve of YOLOv8m after using SAMsaa.

In addition, this paper analyses the convergence curves of two different variants of YOLOv8, i.e., YOLOv8n and YOLOv8m, using the SAMsaa tool accelerated with TensorRT for four performance metrics: precision, recall, mAP@0.5 and mAP0.5:0.951 (Table 2), among other things, the evaluation indicators are mainly used positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

Equation 1: Precision

$$Precision = \frac{TP}{TP + FP}$$

Precision indicates the proportion of correctly recognized apple pixels among all pixels predicted to be apples.

Equation 2: Recall

$$Recall = \frac{TP}{TP + FN}$$

Recall indicates the proportion of correctly recognized apple pixels among all real apple pixels.

Equation 3: Map

$$mAP = \frac{1}{N}\sum_{i=1}^{n}(R_n - R_{n-1})\,p_n$$

R_n and p_n nth Recall and the corresponding Precision value, N is the total number of samples. mAP@0.5 (mean Average Precision, IOU threshold at 0.5): mAP@0.5 represents the detection precision of the model for apples when the Intersection over Union (IOU) threshold is 0.5. It reflects the average precision of the model when the overlap area between the predicted bounding box and the ground truth box reaches 50%. mAP0.5:0.95 (mean Average Precision, IOU threshold from 0.5 to 0.95): mAP0.5:0.95 represents the average of mean Average Precision values across IOU thresholds ranging from 0.5 to 0.95 (with a step size of 0.05). It comprehensively evaluates the model's ability to detect apples at different levels of overlap, reflecting the overall performance of the model.

We hypothesise that the model depth and complexity of each version of YOLOv8 is the key factor leading to the highest accuracy. Following the experimental section above, we performed a statistical analysis of the four model variants, YOLOv8s, YOLOv8n, YOLOv8l and YOLOv8m, after labelling the dataset with the SAMsaa tool and using the TensorRT optimisation technique on the Jetson Xavier NX Edge Computing device and comparing their performances on the aforementioned metrics, as follows (Table 2 and Table 3). By statistically comparing the precision, recall and average precision of each model, it is possible to determine which factors contribute most to the improvement in model performance.

Table 2. Comparative experiments on detection using different variants of the same algorithm.

Method	Recall	Precision %	mAP@0.5%	mAP@0.5:0.9%
YOLOv8n	83.9	89.51	87.45	66.19
YOLOv8m	**98.66**	**97.29**	**97.58**	**83.42**

Table 3. Comparative Experiments in Edge Computing Using the SAMaaa Tool.

Different versions	**Time（ms）**	**TensorRT Time（ms）**
YOLOv8s	35.1	22
YOLOv8n	33.7	17
YOLOv8l	125.6	52
YOLOv8m	**71**	**31**

The results show that the YOLOv8m model outperforms the other model variants on all performance metrics after improving the quality of the dataset through semi-automatic data annotation using the SAMsaa tool, as well as improving the speed and efficiency of the model inference using the Jetson Xavier NX device and TensorRT technology. In particular, the depth and complexity of the model played a key role in the highest accuracy. Specifically, it is shown that: First depth and complexity improvement: the YOLOv8m version of the model has a deeper network structure and more complex feature extraction capability, which can better capture and process complex features in images. In the experiments, the mAP50 of the YOLOv8m model is improved by 33%, and the average detection accuracy of apple detection reaches 90.41%. This result validates the importance of depth and complexity in improving detection performance. Second the impact of high quality data annotation: semi-automatic data annotation using the SAMsaa tool improved the accuracy and consistency of the dataset's annotation, thus providing the models with higher quality training data. This factor has a boosting effect on the performance of all models, but is particularly significant for the YOLOv8m model with higher depth and complexity. Experimental results show a 32.7% improvement on mAP50 for the YOLOv8m version. Finally, the advantage of hardware acceleration: using TensorRT technology for inference acceleration on the Jetson Xavier NX device significantly reduces the computational latency of the models and improves the efficiency of real-time detection. While all models benefit from hardware acceleration, the YOLOv8m model, due to its higher complexity, is able to take fuller advantage of these optimization techniques and thus excels in performance metrics. The final test results showed a high level of accuracy (Figure 7).

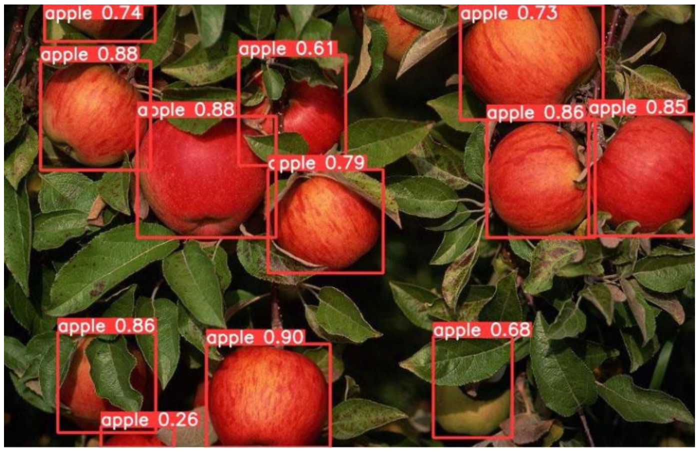

Figure 7. Detection results image.

Conclusion

In conclusion, this study has made significant strides in improving orchard apple detection through the comparison of various YOLOv8 model variants, the application of semi-automatic labeling tools, and the optimization of edge device deployment. Key findings include the superior performance of the YOLOv8m model in balancing accuracy and speed, the efficiency gains from using SAM-based labeling tools, and the substantial improvements in inference speed through TensorRT optimization on Jetson Xavier NX devices.

Acknowledgments

I would like to thank the advisor for the guidance provided.

References

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). Ieee. https://doi.org/10.1109/cvpr.2005.177.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*. https://doi.org/10.1145/3065386.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. https://doi.org/10.48550/arXiv.1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9). https://doi.org/10.1109/cvpr.2015.7298594.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 31, No. 1). https://doi.org/10.1609/aaai.v31i1.11231.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587). https://doi.org/10.1109/cvpr.2014.81.

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, *39*(6), 1137-1149. https://doi.org/10.1109/iccv.2015.169.

Redmon, J. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. https://doi.org/10.1109/iccv.2015.169.

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271). https://doi.org/10.48550/arxiv.1612.08242.

Redmon, J. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. https://doi.org/10.48550/arXiv.1804.02767.