Identifying fruit types and grasping the defective based on facilitated machine learning

Zhuoyang Ding, Xuan Tong, Tian Qiu, Xinshu Hu, Zhiqi Yin, Jingxuan Lu

***Abstract*—This project develops a robot arm taking advantage of facilitated machine learning techniques in identifying fruit types and detecting defects. The ability to accurately classify fruits and identify defects is crucial in various industries, including agriculture, food processing, and quality control. Traditional methods of fruit classification and defect detection often rely on manual inspection, which is time-consuming, subjective, and prone to errors. In recent years, facilitated machine learning algorithms have emerged as a promising solution to automate this process, offering improved accuracy, efficiency, and objectivity. This project used some algorithms involved in fruit type identification and defect detection, such as yolo (You Only Look Once) and ViTs (Vision Transformers). The two algorithms are used in the different stages, and both has a high level of accuracy in identifying the fruits. By combining the algorithms and ROS moveit, we can get a robot arm detecting different types of fruit and defects.**

Key words— Deep learning; Fruit identification; Dregs detection; Agricultural automation; Image processing

I. Introduction

Fruits are an essential component of our diet, providing vital nutrients and contributing to our overall well-being. The demand for high-quality fruits has increased significantly over the years, necessitating efficient techniques for fruit type identification and defect detection. Accurate classification of fruits is crucial for farmers, distributors, and consumers, as it allows for appropriate sorting, grading, and quality control measures. Similarly, the early detection of defects such as bruises, diseases, or deformities is vital to ensure only the best fruits reach the market, reducing waste and maintaining consumer satisfaction.

Traditionally, fruit classification and defect detection have relied on manual inspection by human experts. However, this approach is labor-intensive, subjective, and susceptible to human errors. Moreover, as the volume of fruits being produced and processed continues to grow, manual inspection becomes increasingly impractical and inefficient. To overcome these challenges, facilitated machine learning techniques have emerged as a powerful tool for automating these tasks.

This project aims to develop a comprehensive assembly line that includes conveyor belts, sorting robotic arms, depth cameras, and high-performance core controllers. The conveyor belts will move at a constant speed, transporting fruit to the area beneath the robotic arm's camera. When the depth camera detects fruit within its field of view, it will capture high-speed continuous photos and send the data to an internal controller. The controller will use Vision Transformers (ViTs) to identify defects such as damage, bruising, breakage, and mold spots, and then instruct the robotic arm to remove the defective items into nearby baskets for further processing by staff.

In short, this project has been divided into two parts, one is to use the YOLO and dino-ViTs algorithm to train the model, both of which have reached a high level of accuracy in classifying the fruits, the other is to move the robot arm in a simulation environment.

II. Related Work

A. Neural Networks

With the development of CNNs, image-based machine learning models can be used to make the sorting and grading of agricultural products more efficient[13]. And a a simplified development procedure for image-based machine learning for visual fruit quality assessment has been presented. It is particularly suitable for domains with low availability of both data and computational resources [9].Figure 1gives how vision transform model is different from convolutional neural networks

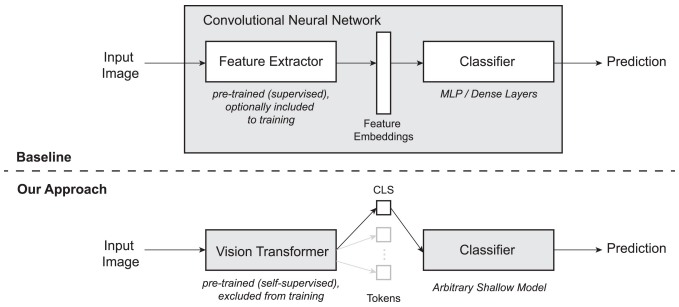


Fig. 1: Vision transformation compared to Convolutional Neural Networks

Based on the generated CLS token, we trained a selection of the most popular shallow machine learning models for classification tasks. The models included in this study are K-nearest Neighbors (KNN), Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), XGBoost (Chen and Guestrin, 2016), and Multilayer Perceptron (MLP), a small neural network. These models are all classical and well-established techniques in the machine learning community and have been shown to perform well on tabular data.

YOLO (You Only Look Once) is a computer vision algorithm based on CNN, the core principle introduced by YOLO involves overlaying a grid of $s \times s$ cells onto the image. It is a real-time object detection algorithm known for its speed and accuracy[6]. Unlike traditional methods that

apply the detection model to an image at multiple locations and scales, YOLO frames object detection as a single regression problem, predicting both the bounding boxes and class probabilities directly from full images in one evaluation. This unified approach allows YOLO to achieve high detection speeds, making it suitable for applications requiring real-time processing.approving the accuracy.

A group of researchers have used the newest version of YOLO, YOLO-v8[7], which has higher accuracy and smaller memory assumption than the previous versions. The group focuses on using the YOLOv8 and CenterNet models to extract visual features from fruit images and analyze fruit peel characteristics to predict the fruit's class. The YOLOv8 model, in particular, incorporates CSP and C2f modules for efficient processing and has achieved an impressive accuracy rate of 99.5% in classifying fruit ripeness[17].

B. Fruit Grasping

When picking fruit, two main considerations come into play. Firstly, fruits come in various shapes and sizes, which can make them difficult to grip. Secondly, most fruits are soft, so using traditional sharp claws can cause damage. However, since most of the fruit we pick are already damaged or unwanted, the first consideration is given more importance.

The article shows a model of a fruit-picking robotic arm[18]. To handle fruits of different shapes, sensors and computer vision are employed to detect and estimate the fruit's position. The robotic arm's inverse kinematics are then calculated based on this position to place the gripper tool in front of the fruit. The final picking method involves iteratively adjusting the vertical and horizontal positions of the gripping tool using a closed-loop visual feedback system[4].

III. Materials and Methods

In this project, the main two technical problems are how to identify our target object and how to plan the trajectory of the arm.

A. Data Sets

Our dataset consists of two parts, which are bananas and apples at different stages of maturity. The banana ripeness data set from Fayoum University (Mazen and Nashat, 2019) contains 273 single fruit images of bananas of different ripeness levels with a neutral background.[1] The researchers who provided this data set labeled it into four ordinal classes: "green" (104 samples), "yellowish-green" (48 samples), "midripen" (88 samples), and "overripen" (33 samples).The apple dataset are from Jahangirnagar University (Nusrat Sultana, Musfika Jahan and Mohammad Shorif Uddin, 2022), including 200 images of fresh apples and 200 images of ripen apples.[2] Since we are using pre-trained models, our data sets require preprocessing to match the resolution that was used for pretraining. The images were downsampled to 224 × 224 pixels keeping the images' original ratio by applying zero-padding to the height dimension (top and bottom side).

[1]https://drive.google.com/drive/folders/1nRWBYAHNRqmL4R0SLrs6dbGQFSWGVY8V?usp=sharing

[2]https://data.mendeley.com/datasets/bdd69gyhv8/1

B. Vision Transformers

Vision Transformers (ViTs) are an innovative image processing algorithm that leverages the transformer architecture initially designed for natural language processing (NLP). The core idea is to divide an image into multiple small patches (e.g., 16x16 pixels) and process these patches as sequences similar to word sequences in NLP tasks. Specifically, the ViT architecture consists of several key components. First, the input image is divided into fixed-size patches, each of which is then flattened into a vector. These vectors are linearly embedded into a higher-dimensional space to form patch embeddings. Since transformers do not inherently capture positional information, positional embeddings are added to these patch embeddings to provide information about the position of each patch within the original image.

Next, the sequence of embedded patches is fed into a standard transformer encoder. The transformer encoder comprises multiple layers, each containing a multi-head self-attention mechanism and a feed-forward neural network. The self-attention mechanism allows the model to weigh the importance of different patches, capturing relationships between distant patches. The feed-forward neural network further processes the data after the attention layer. Layer normalization and residual connections are used to stabilize and improve the training process. Within the encoder, a specific classification token ([CLS] token) is added to the sequence of patches. After passing through the transformer layers, the output corresponding to this token is used for classification through a simple feed-forward neural network. While dk describes the number of key dimensions:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

Regarding training and implementation, ViTs typically employ a pre-training and fine-tuning approach. They are pre-trained on large datasets using self-supervised learning techniques like DINO (self-distillation with no labels), allowing the model to learn useful representations from vast amounts of data without manual labeling. [1]

C. YOLO(You Only Look Once)

YOLO (You Only Look Once) redefines object detection as a regression problem by directly predicting bounding boxes and their associated class probabilities from a complete image. In contrast to traditional object detection methods such as DPM and R-CNN, which typically utilize classifiers for detection and require complex pipelines that are slow and difficult to optimize, YOLO employs a single Convolutional Neural Network (CNN) to enable end-to-end optimization of detection performance[3][5].

YOLO divides the input image into an S×S grid, with each grid cell responsible for detecting objects whose center falls within it. Each grid cell predicts B bounding boxes,

the confidence scores of these boxes, and the conditional class probabilities for C classes. The confidence score reflects the probability of an object being present in the predicted bounding box and the accuracy of the bounding box, defined as Confidence $= P($ Object $) \times \mathrm{IOU}_{\text{pred}}^{\text{truth}}$. The conditional class probability indicates the probability of each class given that an object is present, defined as $Pr(Class_i|Object)$. By multiplying the confidence score with the conditional class probabilities,

$Pr(Class_i|Object) * Pr($ Object $) * IOU_{\text{pred}}^{\text{truth}} = Pr(Class_i) * IOU_{\text{pred}}^{\text{truth}}$

then the final class scores for each bounding box are obtained.

YOLO's network architecture is inspired by GoogLeNet. The standard YOLO model consists of 24 convolutional layers followed by 2 fully connected layers [16]. Unlike GoogLeNet, YOLO uses 1×1 reduction layers and 3×3 convolutional layers [10]. Fast YOLO, on the other hand, reduces the number of convolutional layers and filters, trading some accuracy for a significant increase in processing efficiency.

Training YOLO involves pre-training the convolutional layers on the ImageNet 1000-class competition dataset [15], followed by adapting the model for detection tasks. To enhance performance, YOLO adds four convolutional layers and two fully connected layers with randomly initialized weights and increases the resolution from 224×224 to 448×448. During training, YOLO optimizes a multi-part loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \\
&+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+\sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

This optimization improves the model's stability, preventing divergence during the early stages of training.[12]

D. ROS Moveit

To have a robot arm moving in our simulation environment, we choose to use the ur-5 robot directly, and to have the robot arm moving, ROS[11] and moveit[2] can be a powerful method to complete this.

MoveIt is a state-of-the-art motion planning framework for robotics, designed to facilitate the manipulation, control, and navigation of robotic systems. It integrates a variety of advanced algorithms and tools to provide a comprehensive solution for planning, executing, and visualizing complex robot motions. At its core, MoveIt leverages the Robot Operating System to interface with various sensors and actuators, allowing for real-time perception and feedback. It employs probabilistic roadmap methods (PRM)[8] and rapidly-exploring random trees (RRT)[14] for efficient path planning in high-dimensional spaces, ensuring that robots can navigate around obstacles and optimize their movements. Additionally, MoveIt includes robust tools for kinematics, collision detection, and trajectory optimization[2], making it an essential component for developing sophisticated and reliable robotic applications.

We have been using MoveIt to continuously set the current position of the robotic arm as the initial position and the coordinates of the detected objects on the conveyor belt as the target position. By continuously updating these two positions, the robotic arm has eventually reached the target position, i.e., the position of the block on the conveyor belt, achieving the effect of grasping.

IV. Experiment results

A. Classification results

Using the train_cnn.py script[3], we have effectively trained convolutional neural network models on a dataset comprising images of bananas and apples. Throughout the training process, comprehensive performance monitoring and visualization were conducted via the Weights & Biases (wandb) platform.[4][5]

For the banana training, a total of 240 runs were conducted. We achieved a total of 240 runs by iterating over a combination of multiple parameters. Specifically, we used 10 different seed values ranging from 0 to 9, two operation modes ("all" and "clf"), six different training sample sizes (4, 8, 20, 40, 120, and -1, with -1 signifying all samples), and two models ("resnet50" and "alexnet"). Each unique combination of these parameters constitutes a single run, resulting in a comprehensive exploration across these variable settings. The main purpose of setting 240 runs is to ensure the model's performance consistency and generalization ability under different random seeds. Multiple runs can reveal the overall stability of the model and assess its performance under various initial conditions. This approach provides a more comprehensive understanding of the model's overall performance and reliability. Figure 2 shows the training accuracy and training loss for seeds 0-9. The training accuracy curves for different seeds exhibit a consistent upward trend during the training process, indicating that the model has similar learning ability and convergence patterns under different initialization conditions. Furthermore, the training loss curves for different seeds show a consistent downward trend, with loss decreasing as the training steps increase. This indicates that the loss function optimization is similar across different seeds during the training process.

For the training of apples, to investigate the recognition of healthy and damaged apples, a total of 12 runs were conducted, based on two CNN architectures: VGG and Squeezenet, using a single seed value (0), and utilizing two operation modes ("all" and "clf"). To compare the training effects based on different dataset sizes, the experiment used sample sizes of 50, 100, and 200. Figure 3 shows the recognition results for healthy and damaged apples, demonstrating stable results with

[3]https://github.com/2024Me336Spring/DINO-ViT_fruit_quality_assessment

[4]https://wandb.ai/sustech_me336/dino_baseline_fayoum_reduced_samples

[5]https://api.wandb.ai/links/sustech_me336/fgelglhn

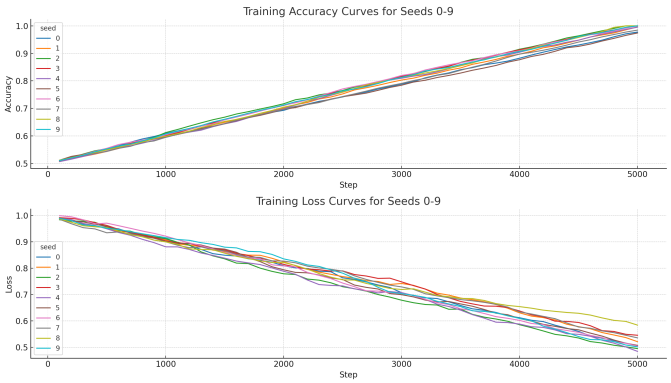


Fig. 2: Training Results Curves for Seed0-9

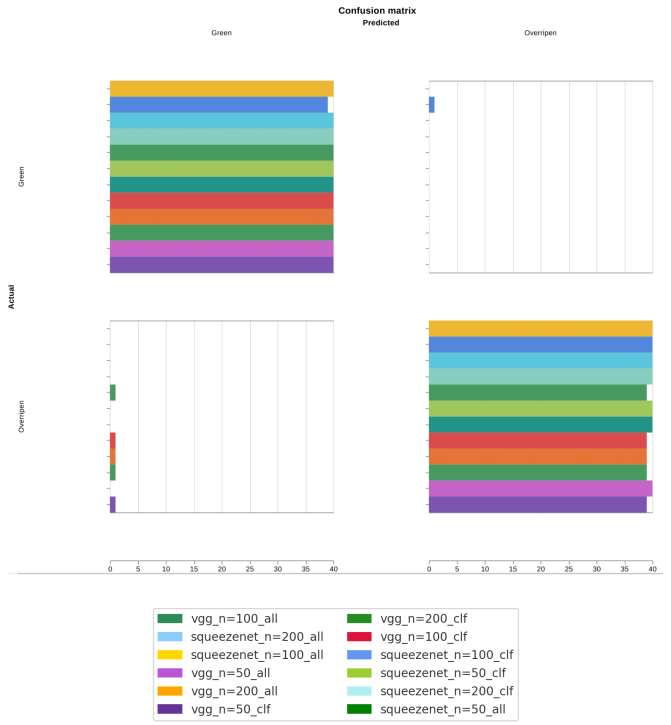


Fig. 3: Confusion Matrix

different CNN architectures and based on different dataset sizes.

The training of the banana and apple datasets confirmed the model's stable performance. The training process is reasonable and exhibits good generalization ability. These characteristics indicate that the model should have high reliability and effectiveness in practical applications.

After evaluating the predictive performance of CNNs and proposing a method for fruit image classification, we compared the best classification accuracy of CNNs with the best performance based on ViTs. Figure 4 shows the results on the banana dataset using shallow classifiers based on two different DINO ViTs (base model and small model). Figure 5 shows a comparison of the results on the banana image dataset between CNN-based and ViT-based methods. In our experimental results, both ViT-based models (SVM classifiers based on DINO ViTB/8 and DINO ViTS/8 embeddings) outperformed CNN-based models. This indicates that pre-trained DINO ViTs can capture domain-specific and local features, and when models are pre-trained on a general dataset, DINO ViT embeddings are generally more accurate at capturing the fundamental features of images compared to supervised CNN embeddings.

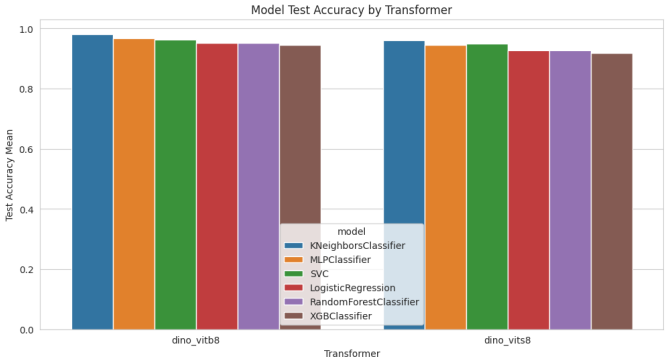


Fig. 4: Model test accuracy by transformer

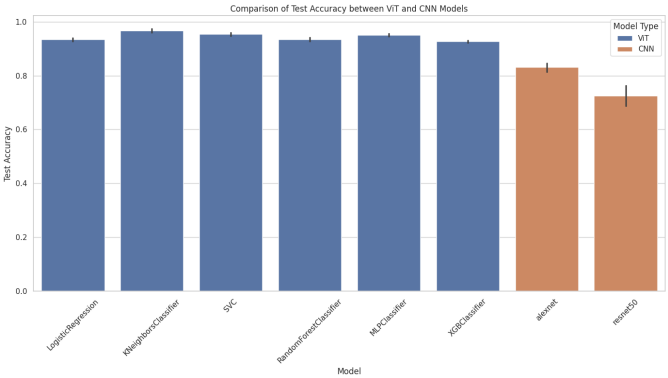


Fig. 5: Comparison between CNN and ViT Models

B. YOLO

In this project, YOLOv8 plays two roles. Initially, during the recognition phase, YOLO functions as a preliminary tool for identifying fruit types. Following this identification, a Convolutional Neural Network (CNN) will use different models according to the various fruit types recognized. YOLO employs its built-in model, yolov8n.pt. The verification method is to put the collected dataset into the model and compare it with the actual situation of the dataset to obtain the accuracy as following figure.6 .

It can be seen that the success rate of the built-in YOLO model in handling the given dataset is not high. In this case, to improve the recognition accuracy, we need to retrain the given dataset.

Training requires the use of a Python application called LabelImg. LabelImg is a graphical image annotation tool. It is written in Python and utilizes Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Additionally, it supports YOLO

and CreateML formats. Using LabelImg, we re-annotate a portion of the fruit images and retrain them according to the fruit types. By employing YOLO's built-in Train function, our team can train the annotated fruit images, resulting in our final model, best.pt. This model is trained entirely on the data dataset and includes only the labels for bananas and apples, thus significantly improving the specificity and accuracy of our model. The Accuracy histogram after training with the new model is shown in figure.7 .

Through specialized training, we can see that the recognition accuracy for nearly all fruits has greatly increased, reaching between 90% and 100%. The training process is illustrated on Wandb, as depicted in the figure.8 . Box_loss, cls_loss, and dfl_loss mean the losses associated with bounding boxes, classification, and distribution focal loss respectively. The reduction in these values indicates model's improvement. Precision and recall denote accuracy and the rate of correctly identified instances. Higher values of these metrics reflect superior model accuracy and performance. The graphical data suggest that the model's indices are improving steadily, with a rapid convergence rate and fine indices, indicating a successful model training.

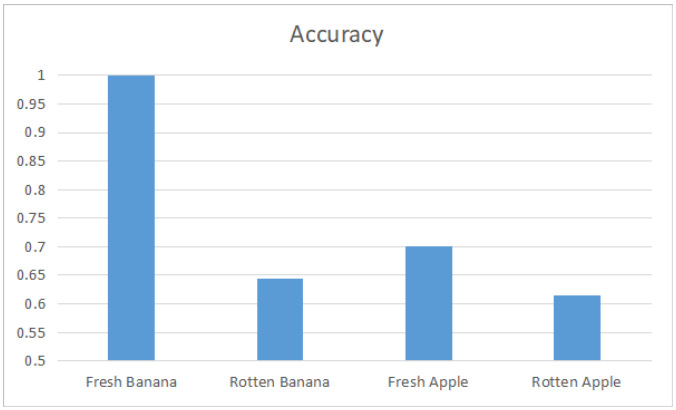


Fig. 6: The accuracy of YOLO's built-in model

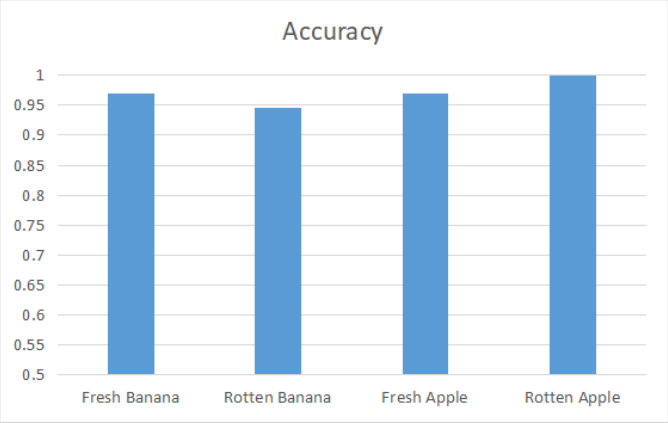


Fig. 7: The accuracy of YOLO's training model

After successfully training YOLO, we can combine YOLO with the subsequent CNN. Following the identification of the fruit by YOLO, the CNN chooses the appropriate model according to the type of fruit identified. The process is shown in the figure.9.

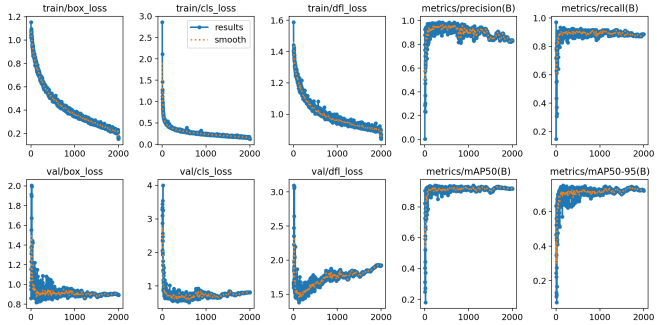


Fig. 8: YOLO's training results

The figure.10 shows the recognition of a bad banana, which will be presented as a successful case.

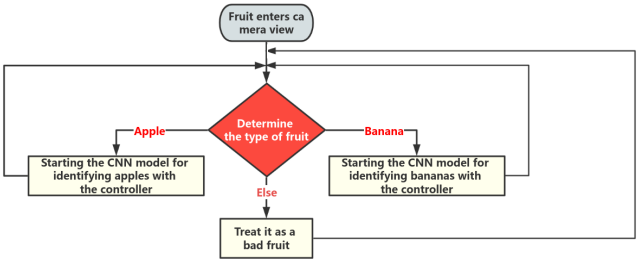


Fig. 9: The Connection Method between YOLO and CNN Networks

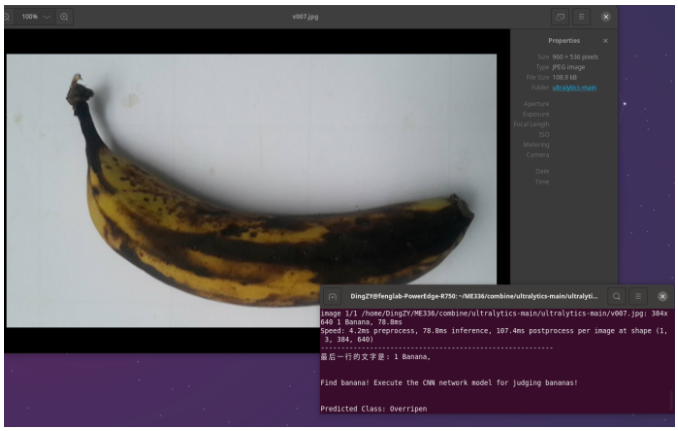


Fig. 10: A successful case

Within the project, the robot arm requires positional calibration of the fruit to perform the grasping task. This is where another function of YOLO, positional calibration, can be employed. YOLO can assist in determining the position of objects, thereby aiding the robot arm in fruit grasping, as shown in the figure.11.

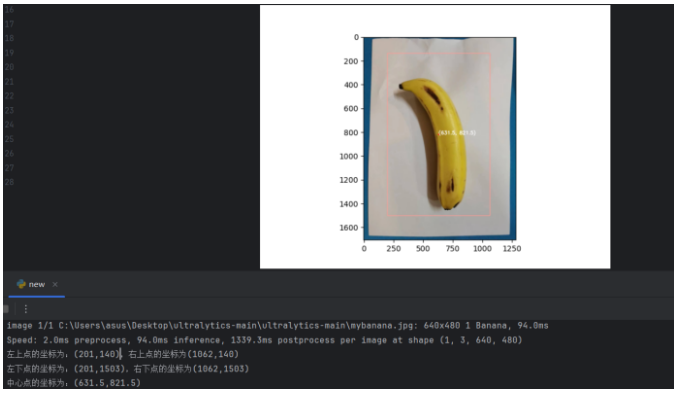
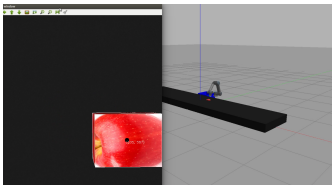


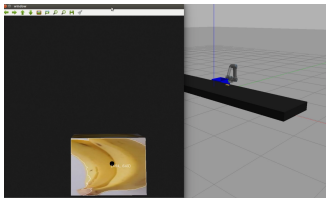
Fig. 11: A practical positional calibration case

C. ROS Simulation

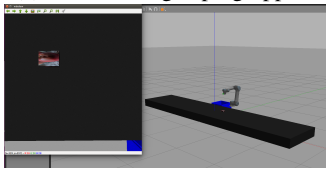
We have used Ubuntu 16.04 and ROS Kinetic as our simulation system. The simulation process is carried out in the Gazebo. We divide the whole activity into two parts: a delaying belt with different blocks moving on it, and a robot arm, which can recognize different figures of fruit, and choose to grasp them. When the robot arm detects the target, we can see a black point in the center of the target, as well as the coordinate of it. Figure 12a and figure 12b shows the robot arm is working to grasp the normal apple and banana, while figure 12c and figure 12d shows the robot arm is letting the inferior apple and banana continuing moving without doing anything.



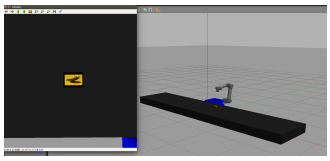
(a) Robot arm grasping apple



(b) Robot arm grasping banana



(c) Robot arm letting rotten apple go



(d) Robot arm letting black banana go

Fig. 12: The working robot arm

V. Conclusion

Our project's goal is to design a mechanized assembly line for sorting fruits, utilizing deep learning networks. We have successfully demonstrated the integration of advanced image recognition technologies and robotic automation to improve the sorting and grading process of fruits with deep learning models. Despite achieving the primary objective, there are still some aspects that require refinement.

Our simulation environment is in ROS. ROS provides a visual platform, which can help us easily identify if the model works well. We have built a convey belt, moving blocks with different figures on them, and also, an ur-5 robot arm, which can distinguish fruits from defective ones, and also, grasp them.

In the first stage of recognition, we used the YOLO. YOLO, serving as the auxiliary algorithm in our project, mainly provides preliminary fruit classification and localization. Regarding fruit classification, although the built-in model has a recognition efficiency of around 70%, the recognition success rate exceeds 95% after specialized training with the collected dataset, effectively classifying fruit types. This enhancement in classification accuracy can significantly improve the efficiency of fruit sorting. Moreover, YOLO's inherent localization function can determine the position of identified objects within a coordinate system, facilitating the robotic arm in fruit grasping.

After completing the fruit classification, our recognition process enters the second phase—fruit sorting through ViT and CNN. Visual Transformers (ViT) represent a novel approach that leverages the powerful capabilities of transformers to process image data. By training ViT on datasets such as apples and bananas, we have validated that it can not only capture complex relationships within images but also achieve exceptional performance even with limited data and resources. Compared to traditional Convolutional Neural Networks (CNNs), ViT excels in capturing long-range dependencies within images and can effectively scale to large networks and datasets. Moreover, ViT performs remarkably well on small datasets, making it particularly suitable for applications that require rapid development and low-cost implementation. After recognition, the robotic arm will sort according to the position provided by YOLO.

Several areas remain for ongoing optimization, including enhancing the performance of ROS and deep learning networks, as well as increasing the realism of simulations. In the future, the group members are willing to apply such a robot arm moving algorithm into a physical robot arm.

Acknowledgments

Thanks to Prof. Song and teaching assistants for the guidance of this project and to every student for their hard work.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[2] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit! [ros topics]. *IEEE Robotics & Automation Magazine*, 19 (1):18–19, March 2012.

[3] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with

discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32 (9):1627–1645, 2009.

[4] Davinia Font, Tomàs Pallejà, Marcel Tresanchez, David Runcan, Javier Moreno, Dani Martínez, Mercè Teixidó, and Jordi Palacín. A proposal for automatic fruit harvesting by combining a low cost stereovision camera and a robotic arm. *Sensors*, 14(7):11557–11579, 2014. ISSN 1424-8220. doi: 10.3390/s140711557. URL https://www.mdpi.com/1424-8220/14/7/11557.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] Muhammad Hussain. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, 11 (7), 2023. ISSN 2075-1702. doi: 10.3390/machines11070677. URL https://www.mdpi.com/2075-1702/11/7/677.

[7] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, January 2023. URL https://github.com/ultralytics/ultralytics.

[8] L.E. Kavraki, M.N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1): 166–171, 1998. doi: 10.1109/70.660866.

[9] Manuel Knott, Fernando Perez-Cruz, and Thijs Defraeye. Facilitated machine learning for image-based fruit quality assessment. *Journal of Food Engineering*, 345:111401, 2023. ISSN 0260-8774. doi: https://doi.org/10.1016/j.jfoodeng.2022.111401. URL https://www.sciencedirect.com/science/article/pii/S0260877422004551.

[10] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. corr abs/1312.4400 (2013). *arXiv preprint arXiv:1312.4400*, 2013.

[11] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, et al. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[13] Ricardo Ribani and Mauricio Marengoni. A survey of transfer learning for convolutional neural networks. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pages 47–57, 2019. doi: 10.1109/SIBGRAPI-T.2019.00010.

[14] Rodriguez, Xinyu Tang, Jyh-Ming Lien, and N.M. Amato. An obstacle-based rapidly-exploring random tree. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 895–900, 2006. doi: 10.1109/ROBOT.2006.1641823.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[17] Bingjie Xiao, Minh Nguyen, and Wei Qi Yan. Fruit ripeness identification using yolov8 model. *Multimedia Tools and Applications*, 83(9):28039–28056, March 2024. ISSN 1573-7721. doi: 10.1007/s11042-023-16570-9. URL https://doi.org/10.1007/s11042-023-16570-9.

[18] Takeshi Yoshida, Yuki Onishi, Takuya Kawahara, and Takanori Fukao. Automated harvesting by a dual-arm fruit harvesting robot. *ROBOMECH journal*, 9(1), 9 2022. doi: 10.1186/s40648-022-00233-9. URL https://doi.org/10.1186/s40648-022-00233-9.