Real Time Cashew Kernel Classification using Deep Learning

Veenadevi S V*, Sowmya Nag K*
Ravikant*, Sagar T Nayak*, Kiran H R†, Manoj Kumar B V†, Yogeesh A S‡
*Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, India
†Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India
‡Department of Biotechnology, RV College of Engineering, Bengaluru, India

***Abstract*—Cashew kernel grading is essential for determining the commercial value of kernels based on shape, size, and surface quality. Traditional manual methods are prone to inconsistencies due to human error, fatigue, and variable lighting, leading to reduced quality and pricing accuracy. This paper presents a real-time automated cashew kernel grading system leveraging computer vision and deep learning. A YOLOv5s model, trained on a custom dataset comprising three commercial grades (W180, W300, W500), was deployed on a Raspberry Pi. The system captures video input via a top-mounted webcam over a conveyor belt, performs kernel classification, and transmits the results to an Arduino Uno through UART. The Arduino controls mechanical flaps via L298N drivers to sort kernels physically. The system achieved over 93% classification accuracy, with physical sorting accuracy ranging from 94–96% across all classes. The proposed hardware-software co-design enables scalable, low-cost deployment in industrial environments, significantly improving throughput and reducing labor dependency.**

***Index Terms*—Cashew kernel classification, deep learning, YOLOv5s, real-time object detection, embedded vision systems, Raspberry Pi, Arduino Uno, conveyor automation, image preprocessing, Roboflow annotation, edge computing, actuator-based sorting.**

I. Introduction

Cashew kernels are one of the most commercially important nuts exported globally, with India, Vietnam, and some African nations dominating processing and trade. Cashew kernel price and quality are mostly defined by their physical traits size, shape, and surface integrity and are utilized to determine standardized grades like W180, W300, and W500 [1]. Traditionally, cashew kernel sorting has been done manually by experienced laborers, but it is subjective, time-consuming, and subject to fatigue-based variations [2].

Recent developments in computer vision (CV) and deep learning (DL) have facilitated effective, automatic examination of farm produce. Object detection models, particularly the YOLO (You Only Look Once) suite of models, have proven to be effective real-time visual classifiers. Lightweight variants such as YOLOv5s are appropriate for edge deployment on resource-limited devices like the Raspberry Pi, facilitating on-device inference at high speed and accuracy [3].

The following is a low-cost, real-time cashew kernel classification system. The solution uses a feeder, a webcam-based image capture setup, a YOLOv5s model on a Raspberry Pi 4, and physical actuation through an Arduino Uno. The custom dataset with images of W180, W300, and W500 kernels was taken under varying illumination, annotated through Roboflow, and augmented for added robustness. The trained YOLOv5s model has a classification accuracy of more than 93% and supports 5 FPS inference. The output of the classification is sent to an Arduino controller that motors a sorting flap with stepper motors and an L298N driver to physically sort the kernels.

The system proposed in this work shows high sorting accuracy, low latency, and robust edge inference support, rendering it suitable for deployment

in small to mid-sized cashew processing.

II. Related Work

Cashew kernel grading automation has gained growing interest, with a view to increasing consistency and throughput in industrial processing. Methods in the early days were based on hand-engineered features and traditional machine learning. The authors of [4], used Support Vector Machines (SVM) to grade cashew kernels based on visual features, but their system was hampered by light and occlusion sensitivity. To enhance robustness, authors of [2], proposed a hybrid solution integrating SVM and Backpropagation Neural Networks (BPNN), although the system did not support real-time hardware integration.

Deep learning networks now replaced conventional kernel classification pipelines. The authors of [5], employed deep CNNs to address lighting and orientation invariance. Subsequently, they developed CashNet-15 [6], an application-specific CNN network trained on augmented data that recorded more than 94% classification accuracies. These were software implementations without deployment on embedded platforms. Authors of [3], suggested a cost-effective embedded CNN-based grading system, but real-time actuation and sorting still remained unsolved.

Object detection models, in specific YOLO, provide a more pragmatic approach. Gautam *et al.* [7], and Raj *et al.* [8], reported that YOLO architectures surpassed CNNs in speed and detection quality for dry fruits and produce. YOLOv5 and YOLOv8 have been found effective in real-time agricultural purposes, with YOLOv8n exhibiting stronger small object detection and anchor-free design [8]. Singh *et al.* [9], introduced SC3T a YOLOv8-based transformer-augmented model hosted on Jetson Nano—registering 97.8% accuracy in real-time grading.

The authors of [10], used ResNet-50 features with SVM-based kernel classification, and Li *et al.* [11], employed a shape-based SVM classifier for detection of whole and split kernels. Bhattacharya *et al.* [12], employed GLCM and color features using Random Forest classifiers for defect detection, whereas Sharma and Ghosh [13], used fuzzy logic for dimensional grading.

Despite these enhancements, many solutions fall short of complete integration with physical sort hardware or rely on high-end computing capability. There are no studies that tackle real-world deployment issues like conveyor sync, lighting fluctuation, or microcontroller integration. This work fills these voids by presenting a whole, low-cost, edge-based system with sorting and classification accuracy for industrial-grade deployment.

III. Problem Statement and Objectives

A. Problem Statement

Cashew kernel grading is very important in establishing market value, export quality, and customer acceptance of processed kernels. Grading has been traditionally conducted manually by experienced operators who classify kernels according to physical attributes like size, shape, and surface integrity. Manual grading is by nature time-consuming, labor-intensive, and subject to inconsistency because of operator fatigue and subjective judgment. These constraints lead to lowered operational efficacy, increased labor expense, and variable product quality.

There remains the widespread adoption of automation technologies across agro-processing businesses, however, that current automated cashew grading devices tend to be too costly, rigid, or demand high-end computing infrastructure inappropriate for SMEs. Additionally, the problem of real-time classification of small, arbitrarily shaped objects such as cashew kernels on resources-constrained edge devices is a major technical challenge. This paper bridges these deficiencies by presenting a low-cost, real-time kernel grading system using a lightweight YOLOv5s model on a Raspberry Pi 4 for cost-effective visual classification, with an Arduino-controlled mechanical sorting system to facilitate autonomous, accurate, and scalable cashew kernel grading.

B. Primary Objectives

The primary goal of this work is to develop an integrated, autonomous cashew kernel classification and sorting system that enhances grading efficiency, consistency, and scalability in small-scale processing units.

The specific objectives of the proposed system are as follows:

- To design a complete grading pipeline that includes image acquisition, real-time classification, and mechanical kernel separation in an affordable, compact, and deployable framework.
- To create a custom image dataset of cashew kernels representing grades W180, W300, and W500 under varied real-world lighting conditions, and to annotate it using Roboflow.
- To train and optimize a YOLOv5s deep learning model for accurate kernel classification, ensuring real-time inference on low-power hardware.
- To deploy the trained model on a Raspberry Pi 4 platform and integrate it with a conveyor-belt mechanism for continuous operation.
- To implement real-time communication between the Raspberry Pi and an Arduino Uno to control stepper motors and actuate a sorting flap for physical segregation of kernels.
- To evaluate the system's performance in terms of classification accuracy, sorting precision, frame rate, and robustness to lighting and positioning variations.

By achieving the above objectives, the system aims to provide a practical, scalable, and cost-effective solution for real-time kernel grading in industrial and SME settings.

IV. Methodology

A. System Architecture

The platform is designed to automate the sorting and classification of cashew kernels with an embedded compact AI pipeline coupled with mechanical parts. As illustrated in Fig. 1, the entire framework is Structured into three main stages that operate within a synchronized real-time loop.

- Visual Data Capture and Input Stream: Raw cashew kernels are dropped from a feeder onto a conveyor belt in a well-spaced, linear flow. A stationary USB webcam placed above the belt takes top-view images under controlled illumination to provide consistent visibility. These real-time images are used as input to the classification module.
- Real-Time Detection and Classification Logic: Processing of captured images is performed using a YOLOv5s model running on a Raspberry Pi 4. The model identifies and classifies kernels into W180, W300, or W500 grades according to shape, size, and surface characteristics. Each detection provides bounding box coordinates and a predicted label that is utilized to control appropriate actuation.
- Grading Execution and Sorting Mechanism: Classification outputs are transmitted over UART to an Arduino Uno that drives stepper motors through an L298N driver. The motors move flaps along the conveyor that physically divert kernels into grade-specific bins. The coordination of actuation and vision processing allows precise real-time sorting in continuous operation.

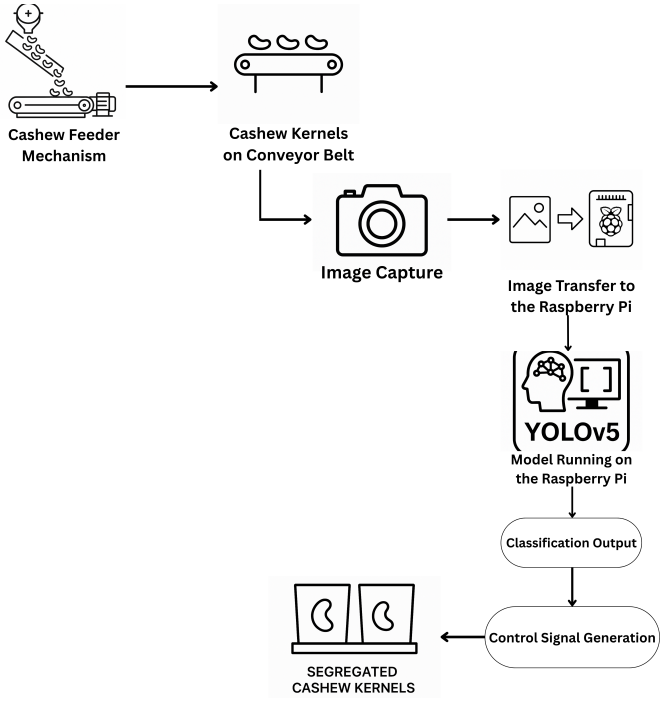


Fig. 1: Real Time System Design of Cashew Kernel Classification

B. System Integration

Approach: The system takes a modular approach that combines deep learning-based computer vision with in-control embedded cashew kernel grading in real-time. A feeder delivers kernels onto the conveyor belt, spaced equally under a USB top-mounted camera. Live frames are executed on a Raspberry Pi 4 that runs an optimized YOLOv5s model for classifying kernels into com-

mercial grades (W180, W300, W500) based on visual characteristics. The anticipated grade is relayed through UART to an Arduino Uno, which controls a motorized flap through an L298N driver to redirect kernels into respective bins. The architecture allows for precise, low-latency operation and is optimized for scalable use in cashew-processing environments. A detailed block diagram of the integrated system is shown in Fig. 2.

Procedures:

1) Cashew Feeder Mechanism: This is the system's beginning. The cashew feeder mechanism releases kernels individually onto the conveyor belt. Spacing is regulated to prevent kernel overlap, which is essential for clean image acquisition and accurate classification. This guarantees individual handling of each kernel in the following processes.
2) Conveyor Belt: As soon as kernels are discharged, the conveyor belt carries them beneath the vision system. The belt speed is adjusted to synchronize with the frame rate and inference delay of the imaging unit, so that each kernel is imaged properly and sorted without bottlenecks.
3) USB Camera – Image Capture: A USB camera is placed above the conveyor and takes live images continuously of the kernels in motion. Image quality is crucial for proper classification. Lighting and camera position should be adjusted to minimize shadows, blur, or distortion.
4) Edge Processing Unit: This part is noted as a subsystem in the diagram and comprises two operations:
   - YOLOv5s Inference – on Raspberry Pi: The Raspberry Pi 4 acts as the edge processor. It executes a trimmed-down version of the YOLOv5s object detection model, optimized to identify various grades of cashew kernels (e.g., W180, W300, W500). The model classifies individual kernels and assigns a category based on characteristics such as size and shape.
   - Classification Logic – Python Script: The YOLO model output is further processed by a Python script. The script reads the detected class and translates it into a designated command or label (e.g., 'W' for whole). The script also formats this data for serial transmission to the actuator control system.
5) UART Communication – Data Transmission: After the kernel has been classified, the Raspberry Pi sends the resulting command to the Arduino Uno using UART (Universal Asynchronous Receiver-Transmitter) serial communication. This provides a quick and efficient handover of classification outcomes to the motor control subsystem.
6) Embedded Control Unit This unit is depicted as an individual subsystem and consists of:
   - Arduino Uno: The Arduino takes the class command from the Raspberry Pi and utilizes it to identify which actuator needs to be activated. It serves as a central sort control hub.
   - L298N Motor Driver: The motor driver converts the low-current GPIO signals from the Arduino to higher-current signals needed to energize stepper motors. It provides for precise direction and speed management, which is imperative to precise flap operation.
7) Actuators: Depending on the signal from the motor driver, the corresponding mechanical flap is triggered. These flaps manually redirect the kernel to its corresponding bin based on its grade. There is a set of flaps for each grade category.
8) Sorted Kernel Bins: The kernels' last destination is a series of bins arranged by kernel grades. After a flap guides a kernel, it falls into the respective bin. The bins enable simple collection, inspection, or packaging for the next phases in the industrial processing chain.

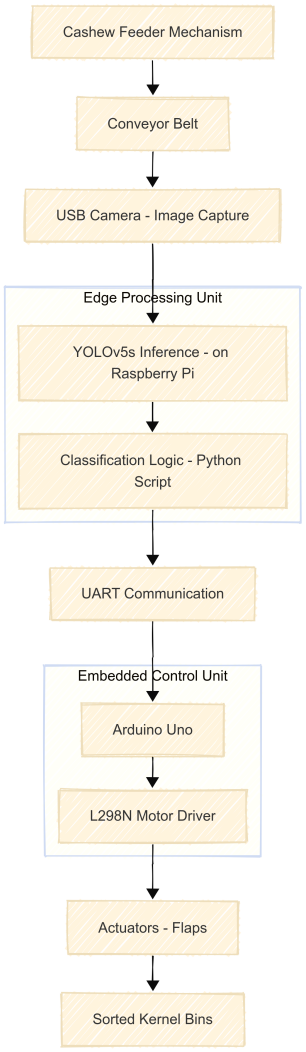
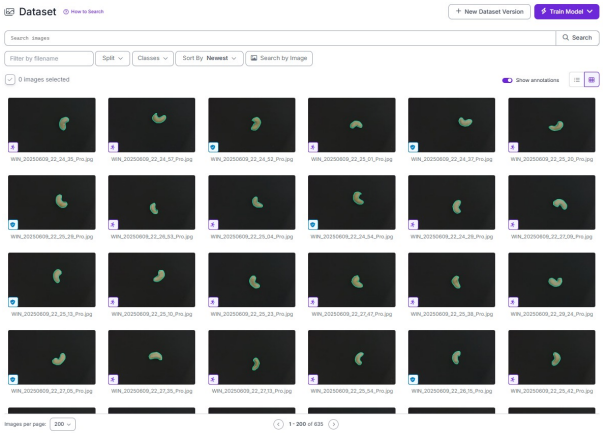


Fig. 2: Detailed block diagram showing component-level integration

C. Dataset Preparation

A personalized dataset of cashew kernels was designed with the help of labeling tools, and thereafter processed on Roboflow for preprocessing, data augmentation, and export in YOLOv5 compatible format. Special care was taken to underrepresented kernel grades (e.g., W180 and W500) to ensure a balanced and diverse training dataset across all categories.



(a) Sample Dataset Images

(b) Annotated Bounding Boxes

Fig. 3: Dataset preparation and annotation process.

D. Model Training and Optimization

The YOLOv5 model was trained on the custom dataset with the following specifications:

- Model variant: YOLOv5s (small) for optimal speed on Raspberry Pi 4 model B
- Training epochs: 100
- Batch size: 16
- Learning rate: 0.01
- Input resolution: 640x640 pixels

1) Central Processing Unit: The Raspberry Pi 4 Model B serves as the central processing unit with the following specifications:

- Processor: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- RAM: 4GB LPDDR4-3200 SDRAM
- Storage: 16GB microSD card for operating system and model storage
- Connectivity: Dual-band 802.11ac wireless LAN, Bluetooth 5.0, Gigabit Ethernet

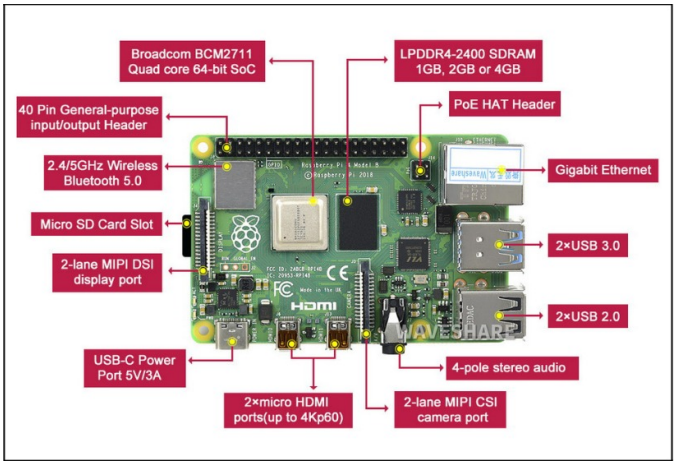


Fig. 4: Raspberry Pi 4 Model B used as the central processing unit.

2) Image Acquisition System: A high-resolution USB webcam module is integrated for real-time image capture:

- Resolution: 1920x1080 pixels (Full HD)
- Frame rate: 30 FPS
- Interface: USB 2.0
- Auto-focus capability for optimal image quality

Fig. 5: USB webcam used for real-time image acquisition.

3) Mechanical Control System: The mechanical system comprises multiple components for physical waste segregation:

- Conveyor Belt: Continuous belt system with adjustable speed control
- Stepper Motors: NEMA 17 stepper motors for precise movement control
- Motor Driver: L298N dual H-bridge motor driver for stepper motor control
- Sorting Mechanism: Motorized flaps/arms for directing waste to appropriate bins
- Power Supply: 12V DC power supply for motor operations

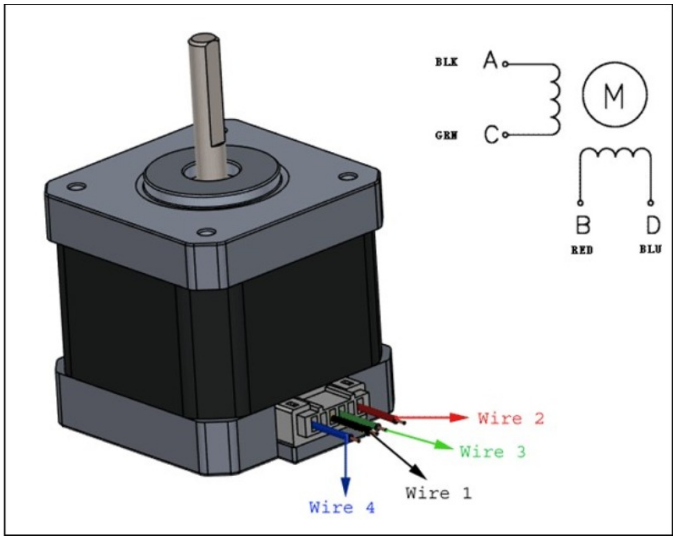


Fig. 6: NEMA 17 stepper motor used for mechanical sorting.

4) Storage Medium: The Raspberry Pi uses a 16GB SDHC (Secure Digital High Capacity) card as its main storage medium. It contains the operating system (usually Raspberry Pi OS), YOLOv5s model weights, Python scripts, and all libraries necessary for image processing, inference, and communication. As a Class 10 SD card, it offers at least 10MB/s minimum sustained write speed, which provides smooth execution of data-hungry processes like logging and model runtime. While small in capacity, 16GB is adequate for lean edge AI applications and ensures quick boot times and reliable runtime performance. It also allows simple replacement and reprogramming during system update or maintenance.

Fig. 7: Secure Digital High Capacity - 16GB

5) Control Interface: Arduino Uno microcontroller handles the low-level motor control:

- Microcontroller: ATmega328P
- Communication: Serial communication with Raspberry Pi
- GPIO pins: 14 digital I/O pins for motor control
- Power: 5V operation with external power supply

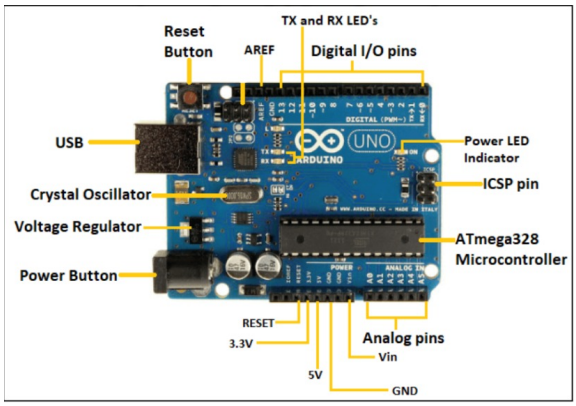


Fig. 8: Arduino Uno used for low-level motor control.

6) Control Communication Architecture: Successful coordination among the classification system and the mechanical sorting unit is achieved through serial communication between Arduino Uno and Raspberry Pi 4. Once the YOLOv5s model recognizes and classifies a cashew kernel, a Python script on the Raspberry Pi sends a class-specific character to the Arduino Uno using UART. On receiving the signal, the Arduino interprets the command and initiates the respective GPIO pin to operate the respective actuator (mechanical flap). This takes place timely and accurately diverting each kernel into its respective bin. The system is real-time with low latency, and rudimentary error-handling logic is incorporated to guarantee error-free communication and actuation.

E. Software Implementation

The software system has been meticulously engineered for real-time kernel classification and accurate actuator control on resource-limited hardware.

1) Deep Learning Framework: The kernel classification is driven by the YOLOv5s model based on the following components:

- Framework: Ultralytics YOLOv5 with PyTorch backend
- Model Optimization: Slim model cut down for speed and performance on Raspberry Pi 4
- Inference Engine: Optimized for ARM architecture with lower latency
- Model Size: Compressed to around 14MB for faster load and execution

2) Computer Vision Pipeline: OpenCV is utilized to manage image acquisition and processing:

- Image Capture: Live frame capture from a USB webcam mounted above the conveyor
- Preprocessing: Resizing, normalization, and channel reordering of YOLO input
- Post-processing: Confidence thresholding and non-maximum suppression to remove duplicate detections
- Visualization: Real-time bounding box and label overlay

3) Control Software: Python scripts control both classification output and hardware coordination:

- Serial Communication: UART is used to send class data to the Arduino Uno using PySerial
- GPIO Control: RPi.GPIO manages auxiliary pins if necessary for lighting or camera control
- Threading: Detection, communication, and monitoring are each handled by separate threads
- Error Handling: Inference and communication error handling and basic error correction logic is implemented

4) System Integration: The system runs in a loop with synchronized hardware-software timing:

- Main Control Loop: Frames are captured, inference is done, and actuators are signaled
- WorkFlow: Cashew enters $\rightarrow$ Classified and detected $\rightarrow$ Signal received $\rightarrow$ Actuator triggered $\rightarrow$ Kernel sorted
- Synchronization: Handoff between Raspberry Pi and Arduino in real-time guarantees no lost kernels

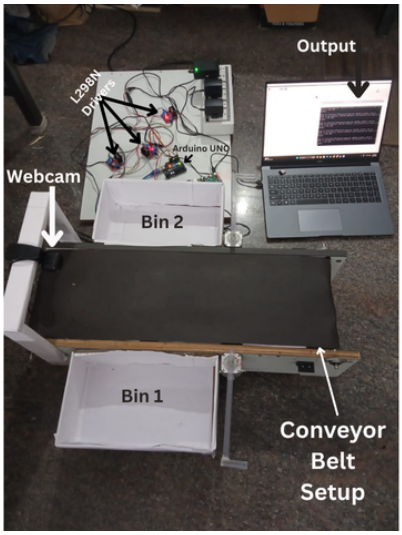
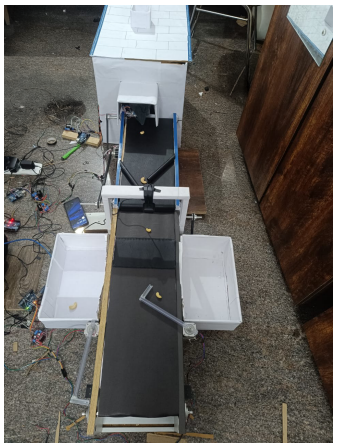


Fig. 9: Initial setup with labeled components

(a) Front View of Hardware Setup



(b) Top View of Hardware Setup

Fig. 10: Real-time prototype implementation showing the hardware layout including feeder, conveyor, camera, and bin segregation

V. Results and Discussion

A. Performance Metrics

The YOLOv5s model was assessed on three whole cashew kernel grades (W180, W300, and W500). The evaluation is based on training performance over 100 epochs, using both qualitative and quantitative metrics. The overall performance is summarized in the following components:

Metric Definitions

1. Training–phase Summary

Table I presents the precision, recall, and mAP@0.5 scores for each class as well as the total dataset. These results reflect consistent classification capability across all kernel grades.

TABLE I: Training–phase performance summary

Class	Total	Correct	Precision	Recall	F1 Score
All	5673	5300	0.936	0.932	0.964
W180	1736	1621	0.936	0.932	0.964
W300	2006	1873	0.936	0.932	0.964
W500	1931	1806	0.936	0.932	0.964

TABLE II: Additional metrics for model performance

Metric	Value
mAP@0.5	0.989

2. Confusion Matrix Analysis

.

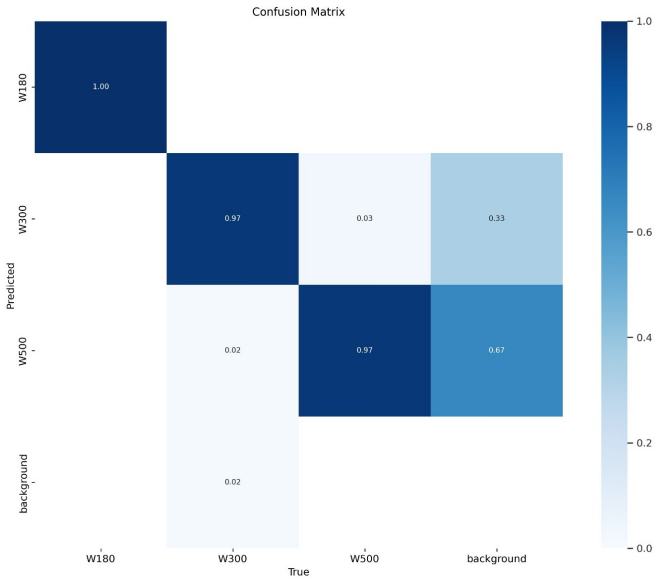
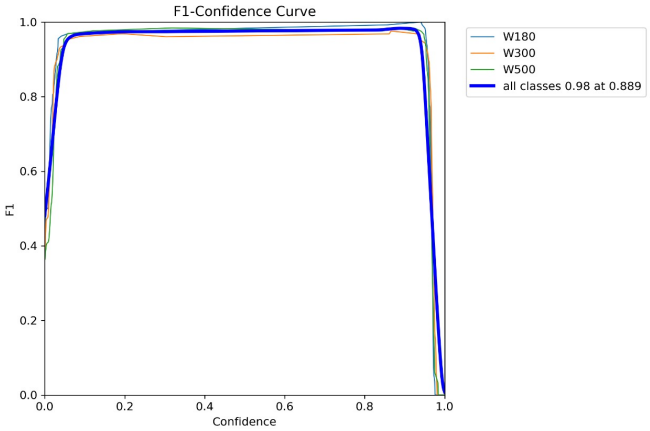


Fig. 11: Normalized confusion matrix showing class-wise distribution of predictions

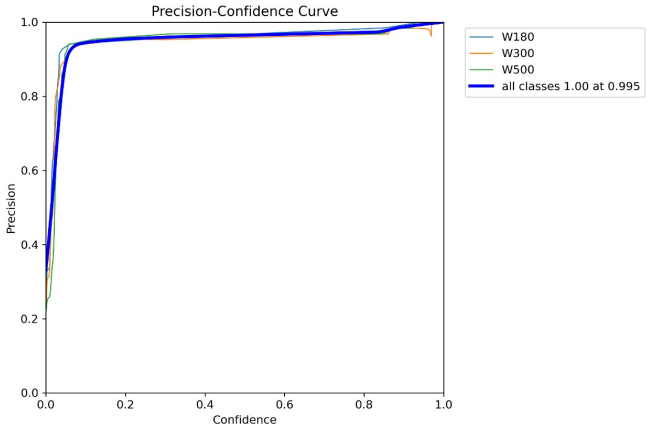
Figure 11 shows excellent class separation with high prediction accuracy. The W180 class achieves perfect classification (100%), while W300 and W500 attain accuracy levels exceeding 94%. Minor off-diagonal elements are attributed to borderline

samples and soft-edged kernels. The results demonstrate reliable kernel discrimination across all three classes

3. F1 and Precision–Confidence Curves



(a) F1–Confidence Curve



(b) Precision–Confidence Curve

Fig. 12: F1 and precision confidence curves across varying confidence thresholds

Figure 12 illustrates how the model's performance varies with confidence thresholds. The F1–Confidence curve peaks at a threshold of 0.89, achieving an F1-score close to 0.98, indicating an optimal balance between precision and recall. Meanwhile, the Precision–Confidence curve demonstrates that the model attains near-perfect precision (almost 1.00) at a threshold of 0.995. These trends confirm that the model is both accurate and dependable, even under high-confidence constraints.

4. Recall–Confidence (RC) Curve

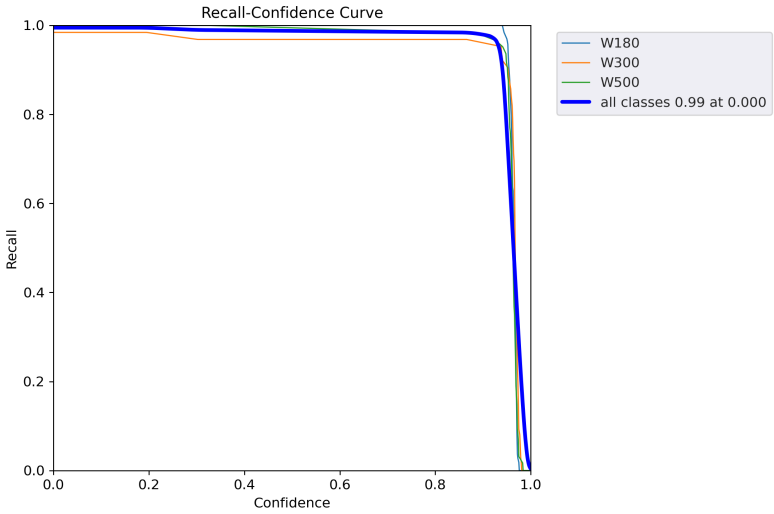


Fig. 13: Recall–Confidence curve showing recall performance across thresholds

The RC curve in Figure 13 illustrates the system's recall performance across varying confidence thresholds. The recall remained consistently above 0.90 for thresholds up to 0.93, indicating that the model maintains high sensitivity and effectively detects most cashew kernels even under moderate confidence requirements. This behavior is crucial for minimizing missed detections in real-time industrial scenarios. However, as the confidence threshold increases beyond 0.93, a gradual decline in recall is observed. This reflects the inherent trade-off between precision and sensitivity—higher thresholds reduce false positives but may lead to missed detections (false negatives). Therefore, selecting an optimal threshold is essential to balance detection reliability and classification confidence in practical deployment.

5. Mean Average Precision

The network attains a mean average precision at 0.5 IoU (mAP_{50}) of **98.9%**. This high score confirms the robustness of the detector in identifying and classifying kernels reliably under varying conditions, making it suitable for real-time automated grading.

6. Predicted Label Visualization

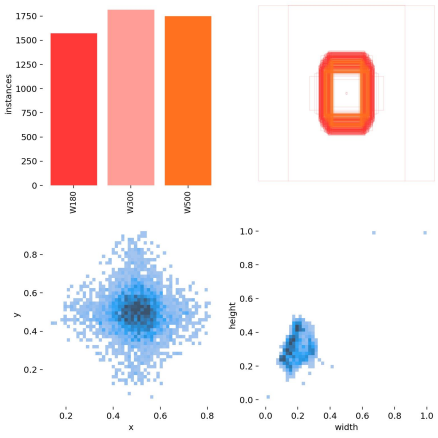


Fig. 14: Label visualization showing bounding boxes and class labels

Figure 14 displays the predicted labels on sample images. The labels are correctly assigned with bounding boxes aligned accurately with kernel contours. This confirms the model's localization and classification performance in real-world test conditions.

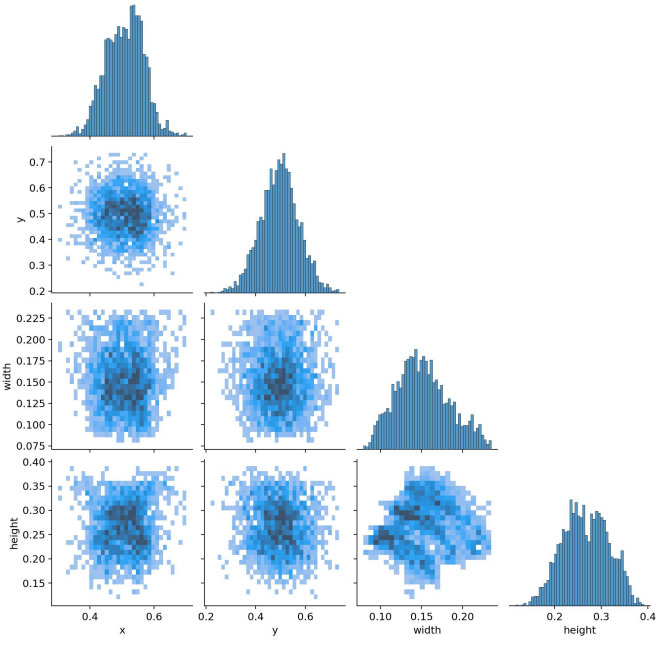


Fig. 15: Pairwise distribution plot of bounding box parameters (x, y, width, height) in the dataset.

Figure 15 illustrates the pairwise relationships between normalized bounding box parameters—center coordinates (x, y), width, and height—used during training. The histograms on the diagonal show that object centers are generally concentrated around the image center, while the scatter plots reveal a uniform spread of object dimensions. This balanced distribution ensures the model sees diverse spatial features during training, aiding better generalization and stability during real-time inference.

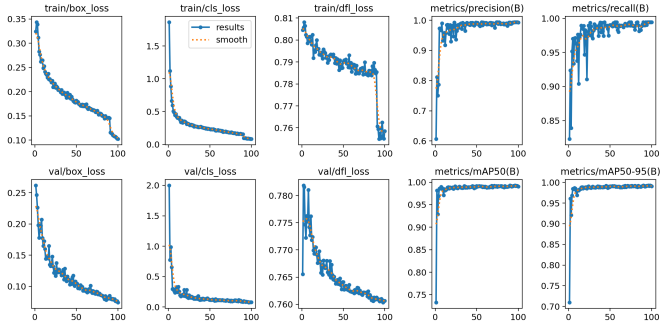
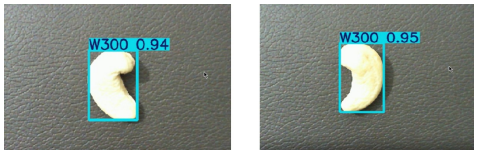


Fig. 16: YOLOv5s training and validation metrics: loss components, precision, recall, and mAP performance over 100 epochs.

Figure 16 displays the evolution of performance metrics during 100 training epochs. The training losses—box loss, classification loss, and distribution focal loss (DFL)—consistently declined, confirming successful convergence. Validation losses followed a similar trend, indicating low overfitting. Notably, the precision and recall values for bounding boxes quickly rose above 0.93 and remained stable. The model achieved a mean average precision (mAP@0.5) close to 0.99 and mAP@0.5:0.95 above 0.96, validating its robust detection performance across varying IoU thresholds. These visual trends further support the strong metrics reported in Tables I and II.

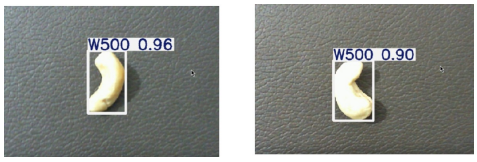
VI. System Validation

Visual inspection on unseen kernels further verified robustness. Figure 17 shows high-confidence detections (> 0.90) for all three grades.



(a) W300

(b) W180



(c) W500

Fig. 17: Inference frames illustrating correct grade prediction with high confidence

A. Evaluation Metric Definitions

Let TP, FP, and FN denote True Positives, False Positives, and False Negatives respectively. The performance of the model is evaluated using the following standard classification metrics:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

The mean Average Precision (mAP) at a specific Intersection over Union (IoU) threshold is computed as:

$$\text{mAP@0.5} = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad \text{where IoU} \geq 0.5 \tag{4}$$

Where N is the total number of classes and AP_i is the Average Precision for the i^{th} class, computed as the area under the precision–recall curve.

The optimal confidence threshold for maximum F1-score can be computed from the F1–Confidence curve as:

$$\hat{\tau} = \arg\max_{\tau \in [0,1]} \text{F1}(\tau) \tag{5}$$

B. System Latency Estimation

The end-to-end latency per classification cycle, denoted as T, is the cumulative sum of the sub-module latencies:

$$T = T_{\text{capture}} + T_{\text{preprocess}} + T_{\text{inference}} + T_{\text{communication}} + T_{\text{actuation}} \tag{6}$$

VII. Quantitative Performance Summary

Table III summarizes system performance across major indicators:

TABLE III: Overall performance metrics of the cashew kernel classification system

Metric	Value
Accuracy	93.4%
Precision	93.6%
Recall	93.2%
F1 Score	96.4%
mAP@0.5	98.9%

A. Limitations and Challenges

- Lighting sensitivity: Degraded recall was observed in early models under unbalanced illumination. Adding the training set with mixed lighting conditions alleviated this problem.
- Kernel overlap: While the detector coped with light overlaps, tightly clustered kernels may still reduce confidence. Future investigations will investigate instance-segmentation or temporal tracking to address heavy occlusion.
- Conveyor drift: Lateral drift during extended runs was small, with a closed-loop belt-position controller in preparation.

B. Discussion

The outcomes verify that a low-weight YOLOv5s network, in conjunction with low-cost embedded platforms, is capable of achieving near-industrial grade accuracy. High accuracy at demanding confidence levels guarantees minimal false diversion, while an overall inference time that satisfies the real-time requirement of small-scale production lines. Mitigating variability in lighting and extreme occlusion will continue to accelerate scalability.

VIII. Future Work

The system developed shows good real-time classification and separation of W180, W300, and W500 cashew kernel grades. There are some improvements that can be made to enhance its scope for classification, robustness in operation, and scalability for deployment on an industrial scale:

- Multi-grade Expansion: The existing model processes three entire kernel grades. The future development will include extension of classification to intermediate and specialty classes like W210, W240, W450, and defective classes like splits, scorched, or immature kernels so that full-spectrum grading can be done according to export standards.
- Dynamic Dataset Generation: For better robustness to background complexity and lighting changes, an automated acquisition module of variable illumination and background texture can be designed. This would enable periodic retraining using new samples for adaptive learning in production environments.
- Edge Acceleration: Raspberry Pi 4, although adequate for present throughput, is limiting under high-load situations. Next-generation versions can consider integration with low-power edge AI accelerators (e.g., NVIDIA Jetson Nano, Coral TPU) to enhance inference speed and enable parallel detection in bulk kernel flows.
- Closed-loop Sorting Feedback: Adding sensors on sorting bins and actuator endpoints can provide a feedback mechanism to ensure flap motions and correct kernel position, improving reliability and minimizing undetected sorting faults.
- Real-time Monitoring Interface: A light sensor dashboard can be created with Python and web technologies to display live classification results, bin status, and model confidence levels—helping operators detect anomalies at runtime.
- Environment-aware Inference: Adding ambient light sensors and adaptive exposure control for the camera module can further reduce misclassification in different lighting environments, particularly in semi-automated processing plants.
- Batch Sorting and Scalability: The system is currently engineered to perform optimally with limited quantities of one or two kernels per frame. Potential future enhancements will incorporate conveyor-based detection and tracking of numerous overlapping kernels in bulk via temporal smoothing and object tracking tools (e.g., Deep SORT).
- Data Logging and Traceability: The system can be designed to record every sorted batch with time stamps, classification breakdown, and kernel images as the platform for traceability and quality control reports for exporters and processors.

IX. Conclusion

This work introduces a complete Deep Learning based cashew kernel grading and separation system that efficiently overcomes the shortcomings of conventional manual grading. The system, combining deep learning, embedded hardware, and real-time automation, provides uniform and precise grading of cashew kernels.

The YOLOv5s-based classifier, running on a Raspberry Pi 4, proves the viability of low-cost, edge-computing technology for industrial processing. The model is over 90% accurate in its detection and provides robust physical segregation via Arduino Uno-controlled stepper motor-operated flaps.

With a scalable, modular, and inexpensive structure, the solution presented herein exhibits great promise for real-world application in cashew processing plants. It increases productivity, grading regularity, and operational efficiency, setting the stage for further optimization and industrial-scale automation in agro-industry use.

Acknowledgment

The authors would like to thank RV College of Engineering, Bengaluru, for providing the necessary resources and infrastructure to conduct this research. Special appreciation goes to the Department of Electronics and Communication Engineering, Department of Computer Science and Engineering and Department of Biotechnology of RV College of

Engineering,Bengaluru, for their continuous support throughout the project development.

REFERENCES

[1] D. Balasubramanian, "Postharvest technology: Physical properties of raw cashew nut," *Journal of Agricultural Engineering Research*, vol. 78, no. 3, pp. 291–297, 2001.

[2] A. Shyna and R. M. George, "Machine vision based real time cashew grading and sorting system using svm and back propagation neural network," in *Proc. Int. Conf. Circuits Power and Computing Technologies (ICCPCT)*, 2017, pp. 1–4.

[3] V.-N. Pham, Q.-H. D. Ba, D.-A. T. Le, Q.-M. Nguyen, D. D. Van, and L. Nguyen, "A low-cost deep-learning-based system for grading cashew nuts," *Computers*, vol. 13, 2024.

[4] A. M. O. Arun, G. N. Aneesh, and A. Shyna, "Automated cashew kernel grading using machine vision," in *Proc. Int. Conf. Next Generation Intelligent Systems (ICNGIS)*, 2018, pp. 1–6.

[5] A. Sivaranjani, S. Senthilrani, B. Ashokumar, and A. S. Murugan, "An improvised algorithm for computer vision based cashew grading system using deep cnn," in *Proc. IEEE Int. Conf. Current Trends in Advanced Computing (ICCTAC)*, 2019, pp. 1–6.

[6] ——, "Cashnet-15: An optimized cashew nut grading using deep cnn and data augmentation," in *Proc. Int. Conf. Systems Computation Automation and Networking (ICSCAN)*, 2019, pp. 1–6.

[7] V. Gautam, R. G. Tiwari, A. Misra, D. Witarsyah, N. K. Trivedi, and A. K. Jain, "Dry fruit classification using deep convolutional neural network trained with transfer learning," in *Proc. Int. Conf. Advancement in Data Science, E-learning and Information System (ICADEIS)*, 2023, pp. 1–6.

[8] R. Raj, S. S. Nagaraj, S. Ritesh, T. A. Thushar, and V. M. Aparanji, "Fruit classification comparison based on cnn and yolo," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1187, p. 012031, 2021.

[9] R. Singh, P. Karthikeyan, and R. Anand, "Sc3t: A low-cost transformer-enhanced deep learning architecture for real-time cashew kernel grading on edge devices," *Computers and Electronics in Agriculture*, vol. 215, p. 108274, 2024.

[10] A. Mishra and S. Awasthi, "Hybrid deep learning and svm model for grading of cashew kernels using resnet-50 features," *Journal of Food Engineering*, vol. 349, p. 111395, 2024.

[11] Y. Li, H. Zhou, and J. Wang, "Lightweight shape-based svm classifier for whole and split cashew kernel detection," *Journal of Imaging*, vol. 9, no. 11, p. 198, 2023.

[12] S. Bhattacharya, S. Roy, and P. Das, "Detection of defective cashew kernels using glcm and color features with machine learning techniques," *International Journal of Food Properties*, vol. 27, no. 1, pp. 154–172, 2024.

[13] V. Sharma and A. Ghosh, "Fuzzy logic-based grading system for cashew kernels using dimensional features," *International Journal of Agricultural and Biological Engineering*, vol. 11, no. 5, pp. 122–129, 2018.