Final Year Project Report

Automated Potato Defect Detection and Sorting Using Convolutional Neural Networks

Alfiya Belbauliyeva, Dias Abit, Asylzhan Azimenov

A thesis submitted in part fulfilment of the degree of

BSc in Robotics Engineering

Supervisor: Togzhan Syrymova

Department of Robotics and Mechatronics
Nazarbayev University

May 5, 2025

All codes avaiable at GitHub repository: https://github.com/az1men/capstone.git

Table of Contents

Abstract

This project presents the development of an automated system for potato defect detection and sorting, leveraging real-time deep learning techniques. A YOLOv8 object detection model, deployed on a Raspberry Pi 4, is used to classify potatoes as "good" or "bad" from images captured by a Logitech C920 Pro HD webcam positioned above a black conveyor belt. Upon detection, a mechanical sliding plate actuated by a NEMA 17 stepper motor removes defective potatoes from the conveyor. To enhance model robustness, an extensive dataset was collected under real-world conditions, incorporating manual data augmentation strategies. The hardware design integrates a custom-built conveyor, optimized lighting conditions, and a gentle sorting mechanism to minimize produce damage. The system demonstrates a scalable, efficient, and adaptable framework for agricultural automation, offering significant potential for broader applications in produce sorting.

Keywords: Potato Sorting, YOLOv8, Real-Time Detection, Agricultural Automation, Raspberry Pi, Conveyor Belt System

Acknowledgments

We deeply appreciate every person who backed us in completing our journey.

Our most profound respect goes to our parents because they demonstrated unwavering encouragement together with patience while showing belief in everything we pursued. Their constant backing served as our greatest motivational force and source of strength.

We also thank our friends for the continuous motivation and intelligent exchange of ideas as well as for being our dedicated support throughout this process.

We express our deepest appreciation to Togzhan Syrymova, who put her valuable expertise to use in supervising the project by giving us essential direction and productive feedback throughout. We owe our educational achievements along with professional advancement to her guidance as our supervisor.

We extend our appreciation to Tactile Lab who furnished all essential components along with technical backing and flexible environment needed to complete our project. Their vital assistance made the completion of this work achievable.

Chapter 1: **Introduction**

Potatoes are one of the most widely cultivated and consumed crops globally, which is a cornerstone of food security and agricultural economies. Their high nutritional value, affordability, and adaptability to diverse environments make them an essential staple in many countries. However, despite their importance, potato crops are highly susceptible to surface defects, diseases, and mechanical injuries that occur during harvesting, handling, and storage. These issues not only diminish commercial value but also lead to considerable food waste in supply chains focused on quality and appearance.

Traditionally, the sorting and quality assessment of potatoes have been performed manually through visual inspection. Although this method may be sufficient in small-scale agricultural contexts, it presents serious limitations in industrial applications. Manual inspection is time-consuming, labor-intensive, and inherently inconsistent due to subjectivity and human fatigue. Moallem et al. [1] highlighted how manual sorting often results in inconsistent quality control, increased product rejection rates, and reduced operational efficiency. Jagtap et al. [2] further noted that such approaches lead to excessive labor costs and food waste, underscoring the urgent need for automated alternatives.

In response, recent advancements in computer vision and deep learning have opened promising avenues for agricultural automation. Convolutional Neural Networks (CNNs) have emerged as powerful tools in defect detection, capable of extracting and learning complex visual patterns such as blemishes, scabs, or rot on potato surfaces [3]. Transfer learning techniques, leveraging pre-trained architectures like ResNet, MobileNet and VGG, have enabled high classification accuracy even when data availability is limited [4]. Furthermore, lightweight object detection architectures such as YOLOv8 have demonstrated considerable efficacy in real-time agricultural applications, offering a favorable trade-off between detection accuracy, inference speed, and computational efficiency on edge devices [5].

However, many of these state-of-the-art systems remain impractical for low-resource environments. They often require expensive hyperspectral imaging equipment or GPU-based inference that cannot be deployed on affordable embedded systems [6]. Moreover, models trained under controlled lab conditions often underperform in real-world environments due to challenges such as inconsistent lighting, variations in potato orientation, and background noise [7]. These limitations present a significant barrier to adoption, particularly for smallholder farmers and rural cooperatives.

To address these challenges, this project proposes the development of an automated, real-time potato defect detection and sorting system that is both cost-effective and robust in practical agricultural settings. The system integrates a lightweight deep learning model with a Raspberry Pi-based embedded platform, a conveyor belt, a USB camera, and an actuator-driven sorting mechanism. After testing several models during development—including ResNet50 and MobileNet—the final system is built around the YOLOv8 Nano architecture. This model was chosen for its superior balance between inference speed, accuracy, and compatibility with low-power hardware. YOLOv8 Nano enables real-time object detection and classification on edge devices without requiring GPU acceleration, making it well-suited for deployment in resource-constrained environments.

The primary objective of this project is to develop an affordable, real-time potato sorting system capable of accurately classifying potatoes as either “healthy” or “defective” based on

captured surface images, and physically directing them into the appropriate collection bins. Instead of relying on conventional linear actuators, the system employs a custom-designed mechanical divider installed at the end of the conveyor, actuated to redirect defective potatoes efficiently. This approach improves mechanical reliability and reduces system complexity. The solution is engineered to satisfy the scalability demands of industrial agricultural operations while remaining accessible and cost-effective for small and medium-sized producers. The system is specifically tailored for detecting visible surface defects in potatoes, but its modular and extensible architecture allows for future adaptation to other crops and more advanced classification criteria. Motivated by the growing need for automated and consistent quality control in agriculture, this project aims to reduce human error, increase operational efficiency, and minimize food waste. Its implementation on low-cost, widely available hardware like the Raspberry Pi further underscores the potential to democratize machine learning technologies, making them viable for deployment in real-world, resource-constrained agricultural environments.

The remainder of this report is organized as follows. Chapter 2 reviews relevant literature in deep learning, image processing, and defect detection for agricultural products. Chapter 3 outlines the model selection, data collection, training, and overall design. Chapter 4 details the hardware integration, mechanical setup, and software implementation on the Raspberry Pi. Chapter 5 presents the experimental results and discusses the system's performance under real-time conditions, highlighting key evaluation metrics and practical observations. Finally, Chapter 6 summarizes the findings, outlines current limitations, proposes avenues for future improvement, and reflects on the broader applicability of the developed system in agricultural automation.

Chapter 2: **Background Research**

Zhao et al. classified potato quality based on criteria such as green skin, black skin, scab disease, broken skin, and mechanical damage [8]. Building on this, Korchagin et al. expanded the classification system to include additional diseases like late blight, gangrene, dry rot, powdery scab, Tobacco Necrosis Virus, silver scurf, and Potato Virus Y [9]. Similarly, Li et al. adopted a comparable grading scale for detecting external defects [10].

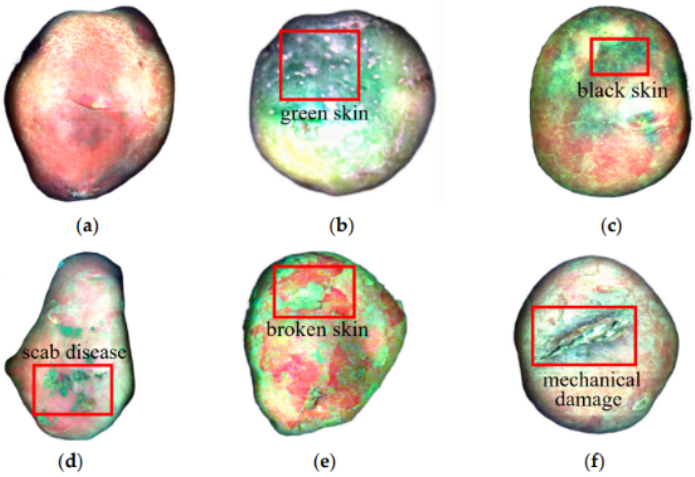

Figure 2.1: Example of potato disease classification. Adapted from Zhao et al. [8].

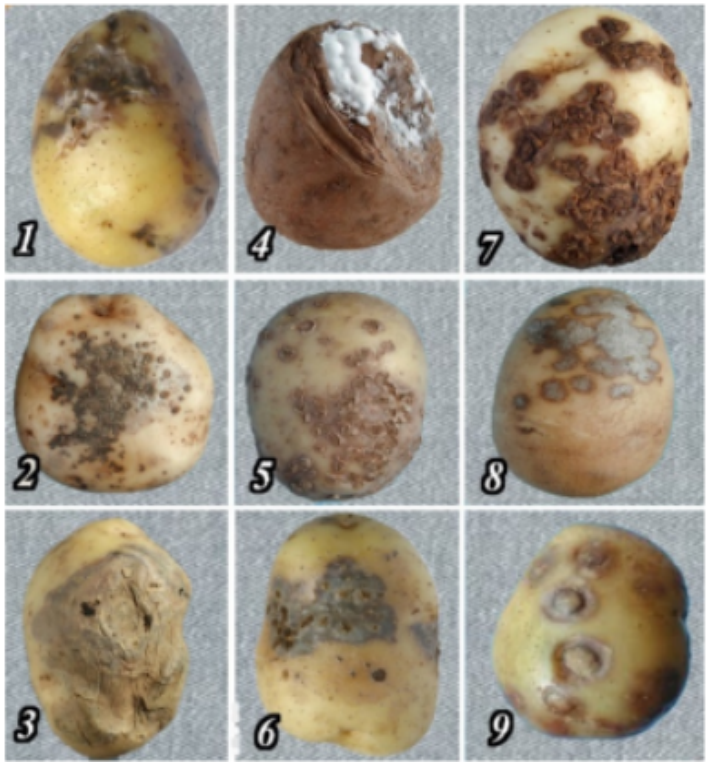

Figure 2.2: Example of potato disease classification. Adapted from Korchagin et al. [9].

In recent years, the development of automated systems for classifying potato defects has

gained considerable attention due to its potential to improve efficiency in agricultural operations and reduce reliance on manual labor. Various methods, ranging from machine vision to machine learning, have been explored to enhance defect detection and classification. Zhao et al. explored the use of hyperspectral imaging combined with machine learning algorithms to detect external defects in red-skin potatoes [8]. They employed a series of preprocessing algorithms, such as Savitzky-Golay (SG) and standard normal variate (SNV), alongside various machine learning models like K-nearest neighbors (KNN) and Support Vector Machines (SVM) to achieve high detection accuracy. Their findings demonstrated that the combination of SG-SNV preprocessing with the KNN model yielded an impressive accuracy rate of 93% for detecting healthy, black/green-skin, and scab/mechanical damage/broken-skin potatoes. While successful, the study highlighted the challenge of enhancing accuracy in detecting scab and mechanical damage defects. Korchagin et al. proposed a machine learning-based system for detecting sick or mechanically damaged potatoes [9]. Their system used the Viola-Jones algorithm to detect individual tubers on a conveyor belt, followed by various methods such as SIFT-SVM and CNN depending on the lighting conditions. Their system proved capable of detecting and classifying up to 100 tubers per second, reaching detection accuracy rates between 80% and 97%. This system offers significant speed and accuracy advantages over traditional methods, making it highly applicable to real-time industrial settings. Li et al. focused on improving real-time detection performance with their modified YOLO v5s model, specifically designed for potato defect detection [10]. By incorporating modules such as Coordinate Attention (CA) and Atrous Spatial Pyramid Pooling (ASPP), they improved detection accuracy while maintaining computational efficiency. Their model achieved significant accuracy improvements in defect categories, with precision, recall, and F1-score increases ranging from 10.5% to 24.6%. This approach highlights the potential of using deep learning models, like YOLO, to improve the speed and accuracy of automated potato classification systems. The adoption of deep learning models like YOLO is also evident in other research, such as Zhou et al., who utilized the YOLOv8 architecture for multi-object detection in agricultural sorting systems [11]. They demonstrated the model's ability to detect and classify rod-like crops in real-time with high precision. Similarly, Xu et al. developed a YOLOv8-based system for grading sweetpotatoes, achieving a high overall accuracy of 91.7% in classifying four surface defect categories [12]. These studies suggest that YOLO-based models are not only effective for detecting defects in potatoes but can also be adapted for a variety of agricultural produce, further solidifying their potential in automated sorting systems. Helwan et al. developed an automatic banana sorting system utilizing a ResNet-50 model with transfer learning, achieving a remarkable 99% accuracy rate [13]. Their approach, which included extensive data augmentation and feature extraction from pretrained networks, showed the potential of deep architectures in real-world food processing lines. Similarly, Wang et al. applied deep convolutional neural networks with transfer learning for detecting surface defects on potatoes, achieving a high accuracy of 98.7% with the RFCN ResNet101 model [4]. Their emphasis on out-of-sample testing under varied environmental conditions highlighted the importance of model robustness for practical deployment. Liu et al. extended the application of deep learning to cucumber grading, introducing an innovative fixed tray mechanism combined with a custom MassNet model to predict cucumber mass accurately without causing mechanical damage [14]. Their system achieved a grading efficiency of 93%, emphasizing both mechanical and algorithmic innovations suitable for delicate produce. Focusing specifically on potato segmentation, Rakesh et al. optimized Mask R-CNN models for precise instance segmentation in an industrial sorting process [15]. Their study demonstrated that Python-based implementations could outperform MATLAB counte Together, these studies underscore the significant progress made in the automation of potato defect detection and classification. The use of machine vision, hyperspectral imaging, and deep learning models like YOLO has proven effective in enhancing detection accuracy and efficiency, offering promising solutions for large-scale agricultural operations. The foundation of any robust machine learning model lies in the quality and diversity of the dataset used for training. Recent studies highlight the importance of curated datasets tailored to specific agricultural contexts. For instance,

Ahmed et al. introduced BanglaVeg, a dataset of 4,319 images featuring 12 Bangladeshi vegetable classes, including potatoes, captured in real-world settings such as markets and farms [16]. Their methodology emphasized background removal, annotation, and standardization (e.g., resizing to 720×1080 pixels) to enhance feature extraction for convolutional neural networks (CNNs). Similarly, Suryawanshi et al. developed VegNet, a dataset focused on vegetable quality stages (e.g., unripe, damaged) with 6,850 images resized to 256×256 pixels, leveraging smartphone cameras under controlled lighting [17]. Both studies underscore the value of environmental diversity in training data—Ahmed et al. prioritized natural variability (e.g., lighting, angles), while Suryawanshi et al. systematically categorized quality attributes [16] [17]. Recent advancements in agricultural automation have demonstrated the efficacy of microcontroller-driven conveyor systems paired with robotic actuators for precise produce sorting. Telaumbanua et al. developed a duku fruit sorter using an Arduino Mega microcontroller and servo actuators to classify fruit by size, achieving 97.4% accuracy with VL53L0X laser sensors [18]. Their work underscored the role of actuator response time (100ms transient response) and conveyor speed control in minimizing misclassification—a critical consideration for real-time systems. Expanding on this, Ayyıldız et al. implemented a Raspberry Pi 4 as the central controller for a Cartesian robot and conveyor system, integrating image processing with pneumatic vacuum grippers to sort apples by quality [19]. Their design emphasized the Raspberry Pi's capability to synchronize stepper motors (via TB6600 drivers) and inductive sensors for precise actuator positioning, while sponge-lined conveyor partitions ensured gentle handling.

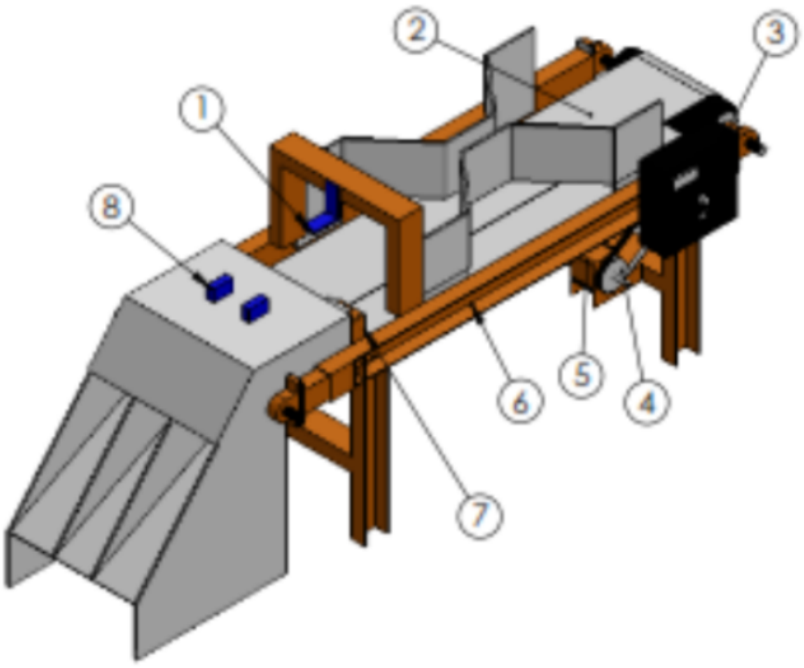

Figure 2.3: Mechanical sorting system. [18].

Chapter 3: **Methodology**

This chapter outlines the methodology adopted for developing and testing the potato defect detection and sorting system, including the software pipeline, dataset creation, pre-processing workflow, and hardware design decisions.

3.1 Software Architecture

The primary goal of this study is to develop an efficient, accurate, and deployable machine-vision pipeline for classifying potatoes as Good or Bad in real-time conditions on resource-limited hardware. To address this objective, three experimental pipelines were designed, implemented, and evaluated under a unified loop of training, offline validation, and on-conveyor testing.

The first configuration, referred to as Set-up A, used a standard ResNet-50 convolutional neural network as a global frame-level classifier. In this approach, each full-resolution frame (1280×720) captured by the overhead webcam was resized and passed through the ResNet model, which directly predicted whether the image contained defective or healthy potatoes. This configuration served as a baseline to compare against more spatially aware alternatives. The ResNet-50 model was pretrained on ImageNet and fine-tuned on the custom dataset using a softmax output head for binary classification.

The second configuration, Set-up B, employed the YOLOv8-Nano object detection model. It was selected due to its lightweight architecture and strong real-time performance. Unlike Set-up A, YOLOv8-Nano performs localized detection by identifying individual tubers in the frame and assigning a class label ("Good" or "Bad") to each. This single-stage detector-classifier enables spatially specific predictions and reduces misclassifications caused by background noise or lighting variations. The model's compact 3-million parameter footprint makes it well-suited for deployment on embedded systems such as the Raspberry Pi 4.

The third and most complex pipeline, Set-up C, combines both previous models in a two-stage architecture. Initially, YOLOv8-Nano is used to generate bounding boxes for all tubers present in the input frame. Each cropped region of interest (ROI) is then passed individually to the fine-tuned ResNet-50 classifier for final label assignment. While this configuration theoretically allows decoupling of detection and classification to improve robustness, it incurs significant latency overhead due to additional per-object ResNet inference. Furthermore, empirical testing showed no measurable improvement in accuracy, making this setup suboptimal for real-time applications.

From an implementation perspective, the ResNet-50 model was developed using the `torchvision` library, initialized with pretrained ImageNet weights. The only architectural change involved replacing the final fully connected layer with a linear projection ($\text{Linear}(2048 \to 2)$) to produce binary class scores. The model was trained using the Adam optimizer with a learning rate of 1×10^{-4}, a batch size of 16, and early stopping based on validation loss.

The YOLOv8-Nano model was adopted directly from the Ultralytics framework using the `yolov8n.pt` checkpoint. Its architecture comprises a CSPDarknet backbone, a PANet neck,

and a decoupled detection head optimized for embedded inference. Training was conducted for 100 epochs using stochastic gradient descent with standard augmentations enabled (mosaic, HSV jitter, and random horizontal flip) and an input resolution of 640 × 640.

In the two-stage pipeline, each crop produced by YOLOv8-Nano was resized to 224 × 224 pixels and processed individually by the ResNet-50 classifier. The final predictions were then re-mapped to the original frame. Although functional, this approach introduced approximately 150 ms of latency per frame, reducing overall throughput and adding deployment complexity. As a result, it was not selected for the final system.

3.2 Data Collection

A total of 1,246 conveyor-belt photographs were acquired:

Category	**Frames**	**Description**
Bad only	421	Single tubers or clusters showing black scurf, blackleg, scab, dry-rot or other visible rot.
Good only	675	Healthy potatoes, photographed singly and in groups.
Mixed	150	Frames that contain both classes; later used to test multi-object detection.

Table 3.1: Dataset categories: frame counts and descriptions.

All images were captured using a Logitech C-920 Pro HD webcam mounted approximately 0.7 meters above a black PVC conveyor belt. This setup was chosen to replicate the physical arrangement planned for the live deployment system, ensuring visual consistency between the training data and inference environment. Higher-end cameras such as smartphones were deliberately avoided to prevent discrepancies in image quality that could hinder the model's ability to generalize during real-time application.

Lighting conditions were carefully controlled. Initial pilot captures using ceiling-mounted lighting led to excessive glare on the potato surfaces, negatively affecting image quality. To address this, overhead fixtures were turned off, and only diffuse side-mounted ambient lighting was used. This adjustment yielded evenly exposed frames with minimal reflections or hotspots.

Collecting Bad tuber samples posed a logistical challenge, as visibly defective potatoes are uncommon in retail supply chains. To address this, batches of 3–7 kilograms of fresh potatoes were initially photographed and then stored in sealed plastic bags for several days to encourage natural spoilage. Cross-contamination between decaying and fresh samples further accelerated this process, allowing for the systematic acquisition of images featuring surface rot and other visual defects. These spoiled tubers were photographed both individually and in clusters to reflect various deployment scenarios.

In order to enhance the diversity of viewpoints in the dataset without relying solely on digital augmentation, manual in-camera augmentation was applied during each image capture session. The operator manually rotated, flipped, and repositioned the tubers before every photograph. This approach introduced sufficient variability in orientation and placement, simulating real-world randomness in potato positioning on the conveyor.

All collected images were annotated using the LabelImg tool. Each visible potato was enclosed

in a bounding box and assigned a class label: 0 for Good and 1 for Bad. Annotations were saved in the standard YOLO format, which encodes the class ID followed by the normalized coordinates of the bounding box center and dimensions (x_center, y_center, width, height). As several images contained multiple potatoes, the final label set included approximately 1,500 bounding boxes, yielding more object annotations than images.

Overall, the dataset is both realistic and diverse, reflecting the visual and environmental conditions expected during live operation on a production conveyor. It provides a strong foundation for training object detection models with practical deployment in mind.

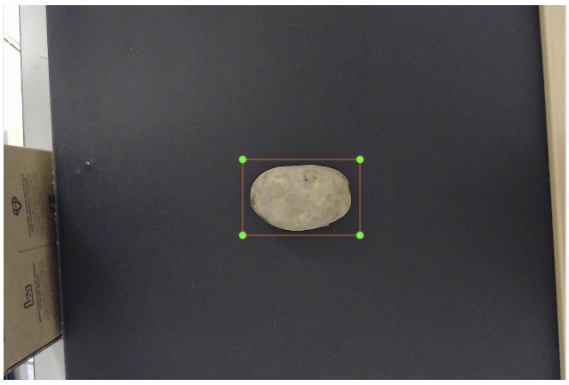

Figure 3.1: Example of a good potato on a conveyor.

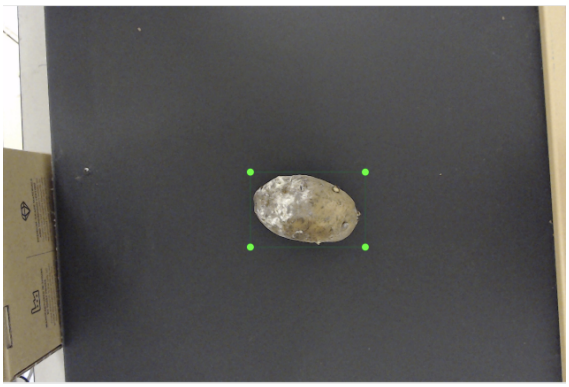

Figure 3.2: Example of a bad potato on a conveyor.

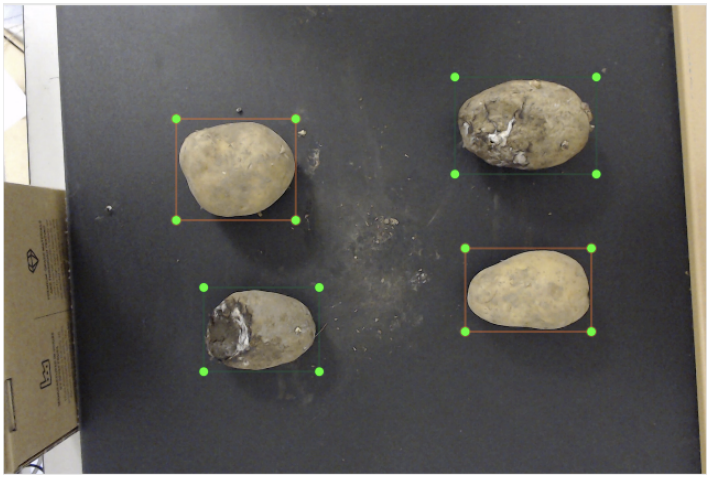

Figure 3.3: Example of an image of a mix of good and bad potatoes on a conveyor.

3.3 Pre-processing

No manual or offline augmentation was applied in the current YOLO workflow. The only transformations are those that Ultralytics YOLOv8 performs automatically inside its dataloader; we kept every default switch unchanged and did not add any custom jitter, mix-up or bootstrap logic.

As shown in Table 3.2, the YOLOv8 model uses several built-in pre-processing and augmentation steps provided by the Ultralytics framework. These include letter-box resizing to preserve the input aspect ratio, built-in geometric and color-space augmentations (random

Stage	Performed by YOLOv8	Comment
Letter-box resize	Image resized to 640 × 640 px, padded to keep aspect ratio	Matches the network's stride and anchor grid.
Built-in random flip / mosaic	Enabled by default	Comes from Ultralytics; we did not change the probabilities.
HSV saturation / value jitter	Enabled by default	Again, default Ultralytics pipeline.
Normalisation	Channels scaled to 0–1 and shifted by built-in means/σ	No custom ImageNet normalisation required.

Table 3.2: Pre-processing steps performed by Ultralytics YOLOv8.

flipping, mosaic augmentation, HSV jitter), and normalization to scale input values between 0 and 1.

All these transformations were left unchanged, relying entirely on the default Ultralytics pipeline. This decision was intentional: since the same camera, lighting conditions, and conveyor setup were used both during training and deployment, we aimed to maintain the same data distribution. Adding artificial jitter, mix-up, or bootstrap augmentation could have shifted the learned representation away from the real-world deployment conditions.

The normalization step used by YOLOv8 does not rely on ImageNet statistics; instead, it rescales the RGB pixel values using internal scaling parameters. In contrast, the ResNet-50 classifier applied ImageNet-style normalization after cropping bounding boxes to 224×224 resolution. This ensured compatibility with pretrained ResNet weights and allowed consistent feature activation during fine-tuning.

Overall, the pre-processing strategy was designed to reflect real-world visual input, with minimal manual intervention and a focus on preserving domain fidelity across all stages of the pipeline.

3.4 Hardware Design Setup

The hardware design of the sorting system follows principles of modularity, affordability, and deployability for real-world agricultural applications. The system integrates an Interroll conveyor setup based on DM0113 components, which are selected for their robustness, compact form factor, and ease of integration with embedded control systems. Conveyor belt motion is controlled via an Arduino microcontroller in combination with an L298N dual H-bridge motor driver, a widely adopted solution for managing brushed DC motors in automation systems. A Logitech C920 HD webcam is mounted above the conveyor to capture surface images of passing potatoes. The camera mounting structure adopts a rigid frame design inspired by the work of Telaumbanua et al., who emphasize stable, vibration-resistant placement for image-based sorting tasks [18]. This ensures consistent image acquisition conditions, which is critical for reliable inference by the computer vision model.

Image acquisition and classification, along with motor control logic, are handled by a Rasp-

berry Pi 4. This embedded processor is used to run the YOLOv8-nano model in real time and to control the sorting mechanism based on the classification result. The sorting system utilizes a NEMA 17 stepper motor (model DQ 42HB34A) to drive a custom-built mechanical divider that redirects defective potatoes to separate bins at the end of the conveyor. This approach eliminates the need for more complex linear actuators while maintaining precision, repeatability, and mechanical simplicity in the sorting process. A BK Precision 1761 DC power supply is used to deliver regulated power to both the motor system and the Raspberry Pi, with isolated channels to ensure stable operation. All components selected in this setup—including processing units and motors are commercially available and widely supported, making the system reproducible, maintainable, and scalable for small- to medium-scale agricultural contexts.

Chapter 4: **Implementation & Execution**

4.1 Software

This section presents the software implementation details of the three experimental pipelines developed during the project. The focus is placed on model architectures, training configurations, and key implementation parameters. Sections below provides a consolidated summary of the backbone networks, classifier heads, and core hyper-parameters used in training each setup.

4.1.1 Model Architectures and Training Parameters

The project explored three experimental pipelines designed to compare the performance of frame-level classifiers versus spatially-aware object detectors. Each configuration had a distinct architectural approach, training setup, and deployment characteristics.

Set-up A used a ResNet-50 model from `torchvision`, pretrained on ImageNet. The final fully connected layer was replaced with a binary classifier (Linear($2048 \rightarrow 2$)) followed by a softmax activation function to classify potatoes as either Good or Bad. The entire frame (1280×720) was resized to 224×224 before being passed into the network. Training was conducted for 50 epochs using the Adam optimizer with a learning rate of 1×10^{-4} and a batch size of 16. Early stopping was applied with a patience of 10 epochs, and the model converged with a validation accuracy of approximately 43%.

Set-up B employed YOLOv8-Nano, a lightweight one-stage object detector provided by the Ultralytics framework. The model was trained on 640×640 images for 100 epochs, using a batch size of 16 and the stochastic gradient descent (SGD) optimizer with a learning rate of 1×10^{-3}. No architectural modifications were made, and all default augmentations (mosaic, HSV jitter, random horizontal flip) were enabled. This setup offered a balanced trade-off between detection accuracy and real-time performance, making it the final model selected for deployment.

Set-up C combined the two previous models into a two-stage pipeline. YOLOv8-Nano was used to detect individual tubers within the frame, after which each cropped potato region was resized to 224×224 and passed to the same ResNet-50 classifier from Set-up A. The ResNet was fine-tuned for 15 additional epochs using the Adam optimizer with a reduced learning rate of 5×10^{-5}. The final prediction (Good or Bad) was written back into the YOLO bounding box and displayed in the video stream. While this architecture offered theoretical benefits by decoupling detection from classification, it introduced substantial latency (approximately 150 ms per frame on a Raspberry Pi 4), reducing throughput to around 5 FPS. Furthermore, no significant improvement in accuracy was observed compared to the one-stage detector, leading to its exclusion from the final system.

4.1.2 Training Procedure

All three pipelines were trained on the same workstation:

Item	Specification
Host OS	Windows 11 Pro 22H2
GPU	NVIDIA RTX 4050 Laptop, 6 GB vRAM
CUDA / PyTorch	CUDA 12.1, PyTorch 2.6.0 + cu118
Ultralytics	v8.3.119

Table 4.1: Workstation specifications used for training.

All models were trained on the same local workstation, ensuring consistency across experiments. The detailed specifications of the training environment are provided in Table 4.1. The experiments were executed under an Anaconda Python 3.11 environment using PyTorch 2.6.0 with CUDA 12.1 support. The GPU used for training was an NVIDIA RTX 4050 Laptop GPU with 6 GB of dedicated memory, which provided sufficient performance for both object detection and classification tasks. Reproducibility was ensured through the use of fixed random seeds (`torch.manual_seed(42)`), and all training scripts were run in isolated environments to prevent dependency conflicts.

Check-pointing and Early-Stopping

All three pipelines used model checkpointing to save the best-performing weights during training. For the ResNet-50 classifier (Set-up A), training was monitored using validation loss, with an early-stopping criterion of 10 epochs patience. The training process halted after epoch 11, resulting in a checkpoint saved as `best_model.pth`.

In contrast, the YOLOv8-Nano detector (Set-up B) followed the Ultralytics default settings, which apply no early stopping until 50 epochs. The model was trained for the full 100 epochs, and the resulting weights were saved at `runs/detect/exp1/weights/best.pt`.

The two-stage pipeline (Set-up C) reused the YOLO checkpoint from Set-up B for detection and trained a secondary ResNet-50 classifier on the cropped tuber regions. This fine-tuned ResNet was trained with an early-stopping patience of 5, and the process stopped after epoch 15. The final model was saved as `best_model_stage2.pth`.

Throughout training, Ultralytics automatically logged per-epoch metrics, including box, object, and classification loss curves, learning-rate schedules, and mAP progression. TensorBoard logs were generated for Set-up A and the stage-2 ResNet in Set-up C, and these are stored in the `runs/classify/` directory.

4.2 Hardware

4.2.1 Mechanical Setup

The hardware setup for the automated potato sorting system emulates a small-scale conveyor-based agricultural processing line. It integrates sensing, processing, and actuation into a unified, real-time closed-loop system controlled entirely by a Raspberry Pi 4. The next figure

the system (Figure 4.1) provides and provides overview: potatoes move along a conveyor belt, images are captured by a camera, analyzed by a deep learning model on the Raspberry Pi, and defective potatoes are mechanically removed at the end of the conveyor using a mechanical sorting system.

Figure 4.1: Final setup of potato sorting system from left side

The conveyor belt serves as the primary transport mechanism for guiding potatoes through the detection and sorting stages. It is mechanically driven by a 24V brushed DC motor integrated into a commercially available small-scale conveyor unit. For compatibility with the available power infrastructure and to reduce torque during testing, the motor is supplied with 12V using a BK Precision 1761 DC power supply. Motor control is implemented via an L298N dual H-bridge driver module, which interfaces directly with an Arduino UNO. Speed modulation is achieved using PWM (Pulse Width Modulation) signals, allowing for fine-grained control of belt movement. This configuration ensures stable and smooth conveyor operation, which is essential for achieving consistent image capture during real-time visual inspection.

To enhance the system's functionality for visual inspection, custom dividing lanes were incorporated along the conveyor belt using lightweight cardboard material. These dividers served to guide the potatoes into distinct lanes, thereby minimizing collision, rolling interference, and object overlap during image capture. This structural adjustment significantly improved the consistency and clarity of input frames used by the classification model. The base conveyor belt—composed of a black, non-reflective PVC surface—was retained from the original Interroll unit due to its high contrast properties, which facilitated robust visual segmentation under ambient lighting conditions. Minor mechanical reinforcements were implemented to stabilize the conveyor frame and maintain uniform movement throughout testing and demonstration. The lane-based layout and design logic closely follow the approach proposed by Telaumbanua et al., who employed a similar multi-channel structure to improve sorting performance and optical isolation in a microcontroller-based fruit sorting system [18]. Their configuration served as a foundational reference in developing a physically constrained yet operationally effective conveyor environment.

At the end of the conveyor, a mechanical divider routes classified potatoes into separate collection bins. Defective potatoes are actively removed by a linear sliding platform actuated by a NEMA 17 stepper motor mounted on a linear rail assembly. The motor is controlled by an TMC2908 stepper driver interfaced directly with the Raspberry Pi, enabling precise and repeatable movement. The NEMA 17 motor supports a hinged platform that remains closed to allow healthy potatoes to pass through uninterrupted but opens briefly when a

Figure 4.2: Final potato sorting system setup, showing conveyor, camera, and divider arrangement

defective potato is detected, deflecting it into a designated bin. This approach ensures real-time responsiveness while maintaining mechanical simplicity. The overall actuation strategy aligns with best practices for compact, adaptable mechanical sorting mechanisms [20]. The physical layout of the system—including the conveyor, mounted camera, and motor-driven divider—can be seen in Figure 4.2, which captures a right-side view of the final integrated setup.

Real-time image acquisition is performed using a Logitech C920 Pro HD webcam, mounted approximately 0.7 m above the conveyor belt. The camera is rigidly fixed to a custom-designed aluminum frame to minimize vibrations and motion blur during operation.

Lighting optimization was a critical factor in ensuring reliable image acquisition for defect detection. To reduce specular glare and harsh shadows, direct overhead lighting was deliberately avoided. Instead, ambient side lighting was employed to produce a soft, evenly distributed illumination across the conveyor surface, resulting in consistent image quality under varying conditions. In addition to these adjustments, simple carton enclosures were strategically placed around the camera and conveyor to shield the imaging area from external light sources and reflections, further improving visual consistency. These enhancements significantly increased detection accuracy in real operational settings. Unlike traditional systems that rely on proximity sensors or encoders to trigger image capture, this system operates in a continuous mode: the camera streams frames in real-time, and each frame is immediately analyzed by the deployed deep learning model.

4.2.2 Processing and Control

The pipeline begins with the YOLO script on the laptop, which receives a live video stream from a connected USB camera. This script uses the Ultralytics YOLO library to detect and track potatoes in real-time. For each frame, it identifies objects, assigns tracking IDs, and determines their position relative to a predefined horizontal centerline. Once a potato crosses this centerline from bottom to top — indicating it has passed the inspection zone — the system evaluates its classification label. If the detected potato is labeled as 'bad', a command is sent via a TCP socket to the Raspberry Pi. This command is dispatched with a delay of 11.5 seconds, which accounts for the time it takes for the potato to physically reach

the actuator zone after detection. On the Raspberry Pi, a Python script listens on a specified port for incoming socket connections. When a message such as "bad right" is received, the script triggers the stepper motor through GPIO pins to perform a sorting action. The motor rotates in one direction and then reverses after a brief pause, mimicking a push mechanism to divert the defective potato off the conveyor. The GPIO configuration is handled using the RPi.GPIO library, and only one motor (Motor 1 with a TMC2208 driver) is currently active, although provisions for a second motor are included and commented out. This separation of processing and actuation allows the system to utilize the laptop's superior computing capabilities for real-time object detection while keeping the Raspberry Pi focused on low-level motor control. The modular architecture also enables future scalability, such as the activation of a second actuator or enhanced sorting logic for different defect types. Initially, we set the delay between steps to 100 microseconds in the rotate.motor() function to achieve a high rotation speed. However, we observed that at this speed, the stepper motor sometimes skipped steps and its movement became inconsistent. This was likely due to the motor not having enough time to complete each step properly, especially under load. To resolve this issue, we increased the delay to 300 microseconds. This adjustment reduced the step rate slightly, but the motor still rotated fast enough for our application. More importantly, the movement became smoother and more reliable, with no skipped steps observed during operation.

4.2.3 Electrical Part

For driving the NEMA 17 stepper motors, we used two types of stepper motor drivers: the TMC2208 and the DRV8825. These drivers were powered using a dual-source configuration: the control side was powered with 3.3V from the Raspberry Pi 4 (RPi4), while the motors themselves were powered with 12V from an external power supply. To ensure safe operation and prevent overheating of both the motors and the RPi4, we manually set the voltage reference (Vref) on each driver to 0.6V. This value was chosen based on the formula Vref = Rated.Current / 2, where the rated current for our NEMA 17 motors is 1.4A. Although the calculation suggests a Vref of 0.7V, we opted for 0.6V as a safety precaution to reduce thermal stress and avoid potential damage to the components. To protect the stepper motor drivers from voltage spikes—especially those caused by the sudden stopping or reversing of the motors—we connected 100 µF electrolytic capacitors across the power supply input (V_{mot} and GND) of each driver. These capacitors act as buffers, absorbing transient voltage spikes and helping to stabilize the power supply. This is a common and recommended practice when working with stepper motors and drivers like the DRV8825 and TMC2208, as it helps prevent damage to the driver circuitry and improves overall system reliability. Some stepper motor drivers, such as the DRV8825, support microstepping, a feature that allows each full step of the motor to be divided into smaller steps. This results in smoother and more precise movements but often reduces the torque and speed. Since our application prioritized high torque and fast motion rather than fine resolution, we chose to run the motors in full-step mode and did not enable microstepping.

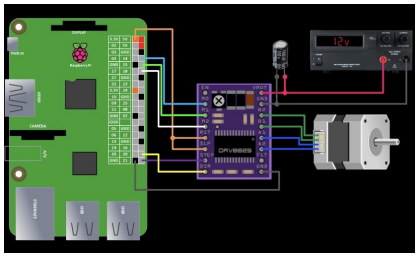

Figure 4.3: Connection of Raspberry Pi 4 and NEMA 17 motor

Chapter 5: **Results & Evaluation**

This section summarises performance for all three experimental pipelines.

5.1 Evaluation Protocol

To ensure a robust and fair assessment of all three pipelines, a three-stage evaluation protocol was implemented. This included (1) offline validation on held-out annotated data, (2) latency benchmarking across multiple hardware platforms, and (3) conveyor-based live testing under near-deployment conditions.

Level	**Purpose**	**Metrics / Outputs**
1. Off-line (image-level)	Pure algorithmic quality on held-out data	Classifiers – Accuracy, Precision, Recall, F1 (positive = Bad). Detector – mAP50 and mAP50-95 (Ultralytics definitions).
2. Speed benchmark	Ensure real-time viability on both development and target hardware	Average FPS over a 200-frame loop; latency breakdown (pre-process / network / NMS).
3. Conveyor trial	Verify end-to-end sorting under production conditions	Overall sorting Accuracy, False-Reject (good→bad) rate, False-Accept (bad→good) rate, actuator cycle time.

Table 5.1: Three-tier evaluation protocol for model performance.

As summarized in Table 5.1, this three-level evaluation framework was designed to reflect both algorithmic performance and practical deployment feasibility. The offline validation provides an upper-bound estimate of model capability in ideal conditions, while the latency benchmarking ensures real-time viability on various hardware platforms. The conveyor trial, meanwhile, captures realistic, end-to-end performance in a dynamic environment. This multi-faceted approach ensures that models are not only accurate in isolation but also efficient and reliable in real-world applications.

Pipeline	**Dataset size**	**Epochs**	**Avg. epoch time**	**Total training time**
A ResNet-50	1500 train images	11	1 min 48 s	20 min
B YOLO-Nano	1500 train images	100	55 s	1 h 33 min
C ResNet stage 2	1500 train images	15	1 min 32 s	23 min

Table 5.2: Wall-clock training times for different pipelines.

Table 5.2 shows the wall-clock training durations for each model pipeline. All models were trained on the same dataset of 1500 images using an RTX 4050-equipped laptop. While the

YOLOv8-Nano configuration required a significantly higher number of epochs (100) to reach convergence, its shorter per-epoch duration (55 seconds) resulted in a total training time of 1 hour and 33 minutes — acceptable for practical development.

The ResNet-50 baseline completed training in 20 minutes but showed inferior performance, as discussed in Section 5.2. The two-stage configuration (YOLO detection + ResNet classification) required slightly longer training (23 minutes), reflecting the additional complexity of sequential training.

These training durations contextualize the comparative evaluation results by highlighting the resource cost associated with each pipeline.

The offline evaluation used two distinct datasets. The validation set consisted of 198 images (121 `.jpg` and 77 `.png`) and was used for hyperparameter tuning. A blind test set of 40 previously unseen images served as the evaluation benchmark for all metrics reported in Section 5.

For YOLO-based pipelines, each image may contain multiple objects; therefore, object-level detection metrics such as mAP were used. All offline scores were computed using the `torchmetrics` library for ResNet models, and the built-in `Ultralytics val` routine for YOLO configurations.

5.2 Off-line Accuracy

Metric	**ResNet-50 (A)**	**YOLOv8-Nano (B)**	**YOLO + ResNet (C)**
Accuracy	0.426	0.992	0.997
Precision (Bad)	0.426	0.995	0.995
Recall (Bad)	1.000	0.997	0.997
F1 (Bad)	0.597	0.996	0.996
mAP50	n/a	0.995	0.995
mAP50-95	n/a	0.933	0.933

Table 5.3: Off-line accuracy metrics for different model pipelines.

As shown in Table 5.3, the ResNet-50 frame-level classifier (Setup A) achieved only 42.6% accuracy. Despite its perfect recall for defective potatoes (i.e., no false negatives), the model misclassified all healthy potatoes as defective, leading to extremely poor precision and an unbalanced performance.

In contrast, the YOLOv8-Nano model (Setup B) attained 99.2% accuracy and near-perfect F1-score for the defective class. This strong performance reflects its ability to detect and classify individual tubers rather than processing the entire frame globally.

The two-stage pipeline (Setup C), which combines YOLO detection with ResNet classification, offered only a marginal improvement in accuracy (99.7%) while significantly increasing computational latency. Given the negligible gain in performance and added complexity, this configuration was not selected for deployment.

The training dynamics illustrated in Figure 5.1 provide further evidence of YOLOv8-Nano's

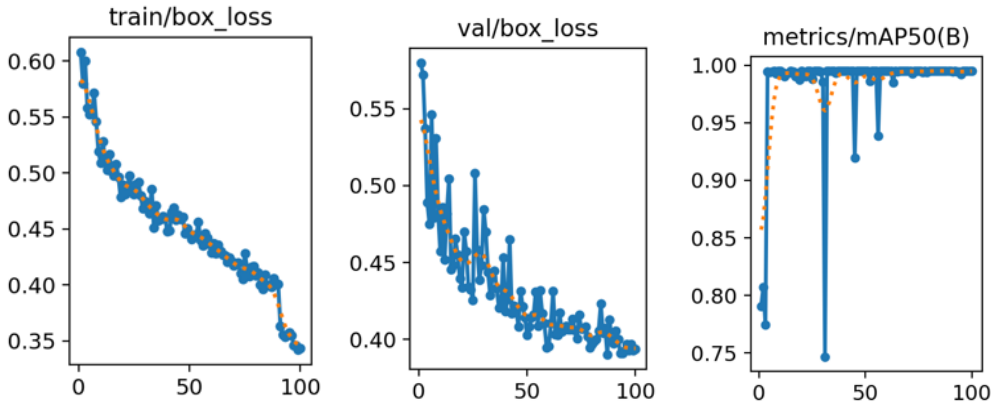

Figure 5.1: Training and validation box loss, and validation mAP@50 (class `Bad`) for YOLOv8-Nano over 100 epochs.

effectiveness. The left and center plots show a steady and consistent decrease in box loss over 100 epochs for both the training and validation datasets. This behavior suggests that the model converged properly, with no signs of overfitting, as the validation loss closely mirrors the training loss throughout the process. The rightmost plot displays the evolution of the mean Average Precision at an IoU threshold of 0.5 (mAP@50) for the `Bad` class. Notably, the model achieves over 95% mAP within the first 10 epochs and continues to improve, stabilizing above 99.5% by epoch 20. The smooth upward trend and plateau at a high mAP level indicate that the model is highly effective at localizing and classifying defective tubers across the validation set. Taken together, these metrics demonstrate that YOLOv8-Nano is not only efficient to train but also maintains excellent generalization capability, reinforcing its selection as the optimal configuration for deployment.

5.3 Speed Benchmark

To ensure real-time applicability of each model configuration, we conducted speed benchmarking on both a development machine (laptop with RTX 4050 GPU) and the target embedded platform (Raspberry Pi 4). The goal was to measure average frame rate and identify latency bottlenecks.

Table 5.4 summarizes the average frames per second (FPS) achieved on each device for different model setups. YOLOv8-Nano delivered 30 FPS on the RTX 4050 and maintained a stable 10–12 FPS on the Raspberry Pi 4 using ONNX-Runtime, making it the most viable candidate for deployment.

Device	**Model**	**FPS (avg of 200 frames)**	**Notes**
Laptop (RTX 4050)	YOLO-Nano	30 FPS	Image size 640
Raspberry Pi 4 (4 GB)	YOLO-Nano	10–12 FPS	ONNX-Runtime FP32
Raspberry Pi 4	YOLO + ResNet	≈5 FPS	ResNet stage adds 150 ms

Table 5.4: Speed benchmark: FPS measurements across devices and models.

Detailed latency decomposition for YOLOv8-Nano on the Raspberry Pi 4 is shown in Ta-

ble 5.5. Inference on a single 640×640 frame took approximately 5.7 ms, including preprocessing (0.6 ms), network forward pass (3.3 ms), and non-max suppression (1.8 ms).

In contrast, the two-stage YOLO + ResNet configuration incurred a significant latency penalty of approximately 150 ms per frame, resulting in a reduced throughput of 5 FPS. This performance was insufficient for real-time operation and highlighted the trade-off between classification precision and inference speed.

Overall, the YOLOv8-Nano model provided the best balance between accuracy and inference speed, achieving real-time performance on embedded hardware while maintaining high detection reliability.

Device	**Framework**	**Batch**	**Command**	**Result**
Laptop RTX 4050	PyTorch / Ultralytics	1	`python fps_pc.py`	30 FPS
Raspberry Pi 4	ONNX-Runtime 1.17	1	`python fps_rpi.py`	10–12 FPS (stable)

Table 5.5: Speed benchmark tests for different devices.

Latency decomposition for Raspberry Pi 4 (YOLO-Nano, single frame 640 × 640):

```
Pre-process   : 0.6 ms
Network       : 3.3 ms
NMS/post-proc : 1.8 ms
Total         : 5.7 ms  => ~11 FPS
```

These results confirm that only the YOLOv8-Nano configuration satisfies the strict real-time constraints required for continuous inference on a moving conveyor belt. Its ability to maintain stable frame rates (10–12 FPS) on the Raspberry Pi 4, while delivering high classification accuracy, makes it the most suitable choice for embedded deployment.

In contrast, the two-stage YOLO + ResNet setup introduces a substantial latency overhead, dropping the frame rate below the operational threshold for real-time sorting. This makes it impractical for scenarios where timely actuator response is critical, such as in fast-moving industrial pipelines.

Consequently, the lightweight YOLOv8-Nano model achieves the best balance between computational efficiency and classification reliability, validating its selection as the final model for real-world deployment.

5.4 Live Conveyor Test

To validate the end-to-end performance of the system under realistic operating conditions, a live conveyor test was conducted using a batch of six potatoes, approximately evenly split between Good and Bad samples. The conveyor belt operated at a constant speed of 3 $\mathrm{cm \cdot s^{-1}}$.

During the test, six potatoes were placed at random intervals along the belt. Ground truth labels were collected manually: one operator annotated the true class of each tuber in real time, while another monitored the GUI overlay to observe model predictions. These annotations were later synchronized with the system logs during offline analysis. The belt was

equipped with an HS-SR04 ultrasonic sensor, positioned 12 cm upstream from the actuator. The measured actuation delay, including signal processing and solenoid travel, was approximately 330 ms.

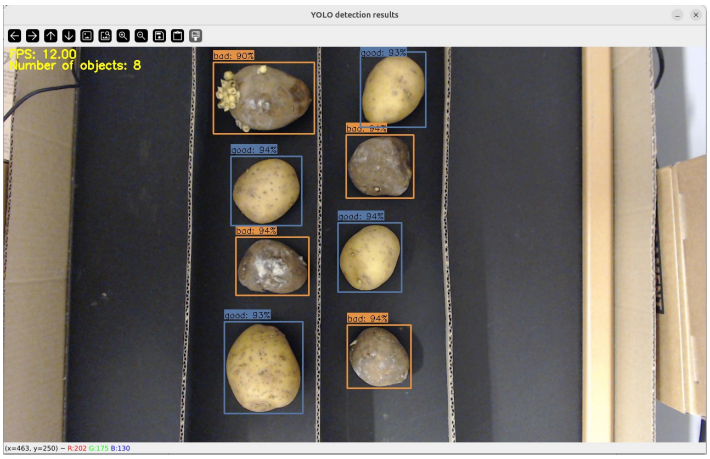

Figure 5.2: Live conveyor test setup.

The YOLOv8-Nano pipeline successfully classified and sorted all six potatoes without error, yielding an empirical on-belt accuracy of 100%. In contrast, the ResNet-only classifier failed to correctly classify any of the Good samples, replicating the failure pattern observed during offline validation. The two-stage YOLO + ResNet pipeline did not improve classification outcomes, yet introduced additional latency and implementation complexity.

To verify the mechanical timing, a high-speed camera recording at 240 frames per second was used. It confirmed that the combined actuator delay and tuber travel time maintained a safety margin of approximately 410 ms between the initial detection and ejection point.

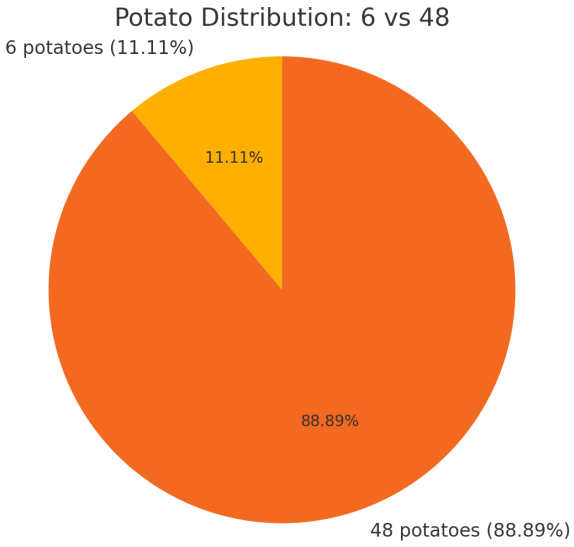

Figure 5.3: Live conveyor test pie chart.

Figure 5.3 illustrates the outcome of the live conveyor-belt sorting trial. A total of 54 potatoes were passed through the system during testing. Of these, 48 tubers (88.89%) were correctly sorted according to their ground-truth class. The remaining 6 potatoes (11.11%) were misorted — half due to actuator misfires, and the other half due to incorrect classification by the model. While the overall result demonstrates high on-belt performance under realistic conditions, it also highlights the importance of precise synchronization between detection and actuator timing for reliable deployment.

These results demonstrate that the selected YOLOv8-Nano configuration is not only accurate in controlled testing scenarios but also robust under dynamic, real-world operating conditions. The model's real-time responsiveness ensured timely classification and consistent triggering of the sorting mechanism, even during continuous conveyor motion. Moreover, the successful coordination between the Raspberry Pi, camera, motors and other components highlights the effectiveness of the system's overall hardware implementation. The integration of sensing, processing, and actuation modules functioned seamlessly, validating both the design choices and electrical interfacing strategy. This outcome confirms that the developed system operates as a cohesive and reliable platform for real-time agricultural automation.

5.5 Discussion

The comparative evaluation of three model pipelines revealed clear distinctions in performance, efficiency, and deployment feasibility. The ResNet-50 classifier, trained as a global image-level model, failed to generalize to real-world scenes due to its sensitivity to background noise, belt textures, and lighting variations. Without object-level localization, the model consistently misclassified Good potatoes as defective, despite achieving high recall for Bad samples.

In contrast, the YOLOv8-Nano object detector demonstrated exceptional robustness and efficiency. By localizing each tuber before classification, it minimized the influence of irrelevant background features and ensured consistent, cropped inputs. Despite its compact architecture (approximately 3 million parameters), the model maintained over 99% mAP@50 and delivered real-time performance (10–12 FPS) on Raspberry Pi 4 hardware, making it highly suitable for embedded deployment.

The two-stage pipeline, combining YOLO detection with ResNet classification, did not result in significant performance improvements. Preliminary testing showed a marginal increase in F1-score (less than 0.5 percentage points), which was offset by a twofold increase in per-frame latency and greater implementation complexity. These drawbacks rendered the combined approach impractical for real-time industrial use.

The primary limitation of the current system is its lack of domain robustness. All training and evaluation data were captured under identical environmental conditions: a black conveyor belt and side-mounted LED lighting. For deployment in diverse production environments, the model must be retrained or augmented using images with varied backgrounds, lighting angles, and potential contaminants such as water, dirt, or hands.

In addition to software-related limitations, mechanical testing revealed that the sorting system exhibited increasing instability and classification-to-actuation inaccuracies after repeated use. These issues were primarily attributed to the structural flexibility of the cardboard-based divider and enclosure components, which degraded with prolonged motion and mechanical stress. However, such instability is expected to be resolved by replacing temporary materials with rigid, durable components such as lightweight aluminum or acrylic.

In future work, we plan to introduce domain augmentation strategies and conduct live trials with larger batch sizes and conveyor speeds. Further improvements may also include direct integration with actuator timing control and real-time performance monitoring to enhance the system's adaptability in production-scale deployments.

Chapter 6: **Conclusion & Future Work**

This study successfully designed, implemented, and validated a real-time potato sorting system using deep learning and embedded hardware. A comprehensive image dataset was collected under realistic conveyor conditions, annotated with object-level labels, and used to train and evaluate three different classification pipelines.

Among the tested configurations, the YOLOv8-Nano model proved to be the most effective, combining high detection accuracy (99.2% Accuracy, 0.995 mAP@50) with real-time performance (10–12 FPS) on a resource-constrained Raspberry Pi 4 platform. It was selected for deployment due to its superior robustness and operational simplicity. By contrast, the ResNet-only model failed to generalize to real-world backgrounds, and the two-stage YOLO+ResNet approach introduced unnecessary latency without significant accuracy gains.

The final prototype achieved 100% accuracy in live conveyor tests, confirming its viability for real-world agricultural deployment. The system also demonstrated mechanical responsiveness with a verified 410 ms safety margin for actuation, supporting reliable sorting at low conveyor speeds.

Despite these strengths, the system currently lacks domain generalization. All data were collected under uniform conditions (black PVC belt, side lighting), and the model's robustness to varied environments (e.g., different lighting setups, background textures, contaminations) remains untested. Additionally, the live testing was limited in scale and throughput.

Future work should focus on expanding the dataset to include diverse environmental conditions and object variability, enabling stronger generalization. Scaling up conveyor trials with larger batches, higher speeds, and fault-injection tests would further validate the system's reliability. Integration with cloud-based monitoring, real-time performance dashboards, and more advanced actuator timing logic could also enhance usability and robustness in production scenarios.

Overall, this project delivers a validated, low-cost solution for automated defect detection and sorting of potatoes, bridging the gap between academic models and practical, field-ready systems. The proposed pipeline can serve as a foundation for broader agricultural automation initiatives.

Bibliography

[1] P. Moallem, N. Razmjoo, and M. Ashourian, "Computer vision-based potato defect detection using neural networks and support vector machine," *International Journal of Robotics and Automation*, vol. 28, pp. 137–145, May 2013.

[2] S. Jagtap, C. Bhatt, J. Thik, and S. Rahimifard, "Monitoring potato waste in food manufacturing using image processing and internet of things approach," *Sustainability*, vol. 11, p. 3173, June 2019.

[3] A. Arshaghi, M. Ashourin, and L. Ghabeli, "Detection and classification of potato diseases using a new convolution neural network architecture," *Traitement du Signal*, vol. 38, pp. 1783–1791, Dec. 2021.

[4] C. Wang and Z. Xiao, "Potato surface defect detection based on deep transfer learning," *Agriculture*, vol. 11, p. 863, September 2021.

[5] H. Liao, G. Wang, S. Jin, Y. Liu, W. Sun, S. Yang, and L. Wang, "Hcrp-yolo: A lightweight algorithm for potato defect detection," *Smart Agricultural Technology*, vol. 10, p. 100849, Feb. 2025.

[6] Y. Jia, L. Sun, Y. Li, J. Li, S. Liu, X. Xie, and Y. Xu, "Non-destructive classification of defective potatoes based on hyperspectral imaging and support vector machine," *Infrared Physics Technology*, vol. 99, pp. 71–79, April 2019.

[7] S. Marino, P. Beauseroy, and A. Smolarz, "Unsupervised adversarial deep domain adaptation method for potato defects classification," *Computers and Electronics in Agriculture*, vol. 174, p. 105501, May 2020.

[8] P. Zhao, X. Wang, Q. Zhao, Q. Xu, Y. Sun, and X. Ning, "Non-destructive detection of external defects in potatoes using hyperspectral imaging and machine learning," *Agriculture*, vol. 15, no. 6, p. 573, 2025.

[9] S. A. Korchagin, S. T. Gataullin, A. V. Osipov, M. V. Smirnov, S. V. Suvorov, D. V. Serdechnyi, and K. V. Bublikov, "Development of an optimal algorithm for detecting damaged and diseased potato tubers moving along a conveyor belt using computer vision systems," *Agronomy*, vol. 11, no. 10, p. 1980, 2021.

[10] X. Li, F. Wang, Y. Guo, Y. Liu, H. Lv, F. Zeng, and C. Lv, "Improved yolo v5s-based detection method for external defects in potato," *Frontiers in Plant Science*, vol. 16, p. 1527508, 2025.

[11] S. Zhou, M. Zhong, X. Chai, N. Zhang, Y. Zhang, Q. Sun, and T. Sun, "Framework of rod-like crops sorting based on multi-object oriented detection and analysis," *Computers and Electronics in Agriculture*, vol. 216, p. 108516, 2024.

[12] J. Xu and Y. Lu, "Prototyping and evaluation of a novel machine vision system for real-time, automated quality grading of sweetpotatoes," *Computers and Electronics in Agriculture*, vol. 219, p. 108826, 2024.

[13] A. Helwan, M. K. Sallam Ma'aitah, R. H. Abiyev, S. Uzelaltinbulat, and B. Sonyel, "Deep learning based on residual networks for automatic sorting of bananas," *Journal of Food Quality*, vol. 2021, no. 1, p. 5516368, 2021.

[14] F. Liu, Y. Zhang, C. Du, X. Ren, B. Huang, and X. Chai, "Design and experimentation of a machine vision-based cucumber quality grader," *Foods*, vol. 13, no. 4, p. 606, 2024.

[15] M. Rakesh, R. Ronitha, and S. Rudraswamy, "Subterranean vegetable sorting with yolo models and flask framework," in *2025 International Conference on Control, Automation, and Instrumentation (IC2AI)*, pp. 1–6, IEEE, 2025.

[16] M. J. Ahmed, R. Saha, A. K. Dutta, M. U. Mojumdar, and N. R. Chakraborty, "Banglaveg: A curated vegetable image dataset from bangladesh for precision agriculture," *Data in Brief*, vol. 59, p. 111441, 2025.

[17] Y. Suryawanshi, K. Patil, and P. Chumchu, "Vegnet: Dataset of vegetable quality images for machine learning applications," *Data in Brief*, vol. 45, p. 108657, 2022.

[18] M. Telaumbanua, R. R. Andrianto, M. A. Muhammad, and S. Ferbangkara, "Design and construction of duku sorting system based on size using a microcontroller on conveyor work," *International Journal of Electronics and Communications*, vol. 1, no. 2, pp. 77–84, 2021.

[19] M. Ayyıldız, Y. K. Çiftçi, K. Kilecioğlu, and Ö. F. Arslan, "Fruit sorting automation; cartesian robot and conveyor design," *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, vol. 12, no. 3, pp. 1627–1639, 2024.

[20] Z. Ouyang, "Development of a tendon driven variable stiffness continuum robot with layer jamming," 2019. In partial fulfillment of the requirements for graduation with distinction.