Fruit Quality Detection Using Jetson Nano

Krishnansh Vasaniya
MIT ADT University, Pune
krishnikam92@gmail.com

Arin Shah
MIT ADT University, Pune
arinshah31@gmail.com

Pruthviraj Jagdale
MIT ADT University, Pune
prujag123@gmail.com

Abstract:- **This report presents the development of a real-time fruit quality detection system using the NVIDIA Jetson Nano and YOLOv5. The system captures images of fruits in real-time using a camera and classifies them as fresh or rotten. Based on the classification, the system directs the fruits to different paths. The implementation ensures high accuracy and efficiency, making it suitable for industrial applications such as automated quality control and sorting. This project also explores the integration of hardware components and software optimizations for real-time deployment on edge devices. The pre-trained YOLOv5 model was fine-tuned on a custom dataset of fruit images to achieve optimal performance. Detailed insights into dataset preparation, model training, deployment, and evaluation are provided.**

Keywords:- **Jetson Nano, YOLOv5, Fruit Quality Detection, Real-Time Processing, Edge AI, Automated Sorting.**

I. INTRODUCTION

Fruit quality detection is a critical component of automated food processing industries. Traditional manual inspection methods are time-consuming, subjective, and prone to errors. These limitations can be addressed by leveraging advancements in artificial intelligence (AI) and edge computing. Real-time automated systems can ensure faster and more reliable quality control, reducing operational costs and improving efficiency.

The NVIDIA Jetson Nano, a compact and low-power AI computing platform, offers a cost-effective solution for deploying deep learning models at the edge. Coupled with YOLOv5, a state-of-the-art object detection framework known for its speed and accuracy, this project aims to develop a robust system for identifying and sorting fruits based on their quality. The integration of hardware components such as cameras, actuators, and LED indicators further enhances the practicality of the system for industrial deployment.

The specific objectives of this work include:

1. **Integration of YOLOv5 with Jetson Nano:** Develop an efficient and accurate fruit classification system optimized for edge AI deployment, minimizing latency and maximizing accuracy.
2. **Real-Time Processing:** Configure the system to process video feeds seamlessly and classify fruits in real time, maintaining high throughput and minimal delay.
3. **Performance Optimization:** Ensure the system operates efficiently by evaluating hardware utilization, classification accuracy, and inference latency, with a focus on practical scalability.

The practical applications of this work include:

- **Food Processing Industry:** Automating the sorting process to enhance productivity and reduce manual labor costs in quality control.
- **Supply Chain Management:** Ensuring consistent fruit quality during packaging and distribution, minimizing waste.
- **Agricultural Support:** Providing a scalable solution for on-field sorting during harvest to assist farmers and suppliers.

The experimental validation of this system demonstrates the feasibility of using edge AI to automate and optimize fruit quality detection, addressing challenges like real-time processing and resource constraints effectively.

II. METHODOLOGY

This section outlines the step-by-step process for implementing and deploying the real-time fruit quality detection system on the NVIDIA Jetson Nano using YOLOv5.

A. Hardware and Software Setup

Hardware:

1. **NVIDIA Jetson Nano:** Selected for its energy efficiency and powerful GPU capabilities, enabling real-time AI inference at the edge.
2. **Camera:** A USB camera captures high-resolution images of fruits on a conveyor belt in real time, ensuring optimal visibility for classification.
3. **Sorting Mechanism:** Actuators are employed to direct fruits into appropriate categories based on the YOLOv5 model's predictions.
4. **LED Indicators:** Red LEDs signify rotten fruits, while green LEDs indicate fresh fruits, offering a quick visual summary of the classification process.
5. **Power Supply:** Adequate power ensures smooth operation of the Jetson Nano and connected peripherals.

Software:

1. **YOLOv5 Framework:** Known for its lightweight architecture and accuracy, YOLOv5 is used for object detection and classification of fruits.
2. **PyTorch:** Supports model training and deployment tasks.
3. **JetPack SDK:** Includes essential tools like CUDA, cuDNN, and TensorRT, optimized for the Jetson Nano.
4. **OpenCV:** Facilitates image and video stream handling.
5. **Python:** Serves as the programming language for implementing the inference and control logic.

B. YOLO Model Preparation

Dataset Acquisition and Annotation:

- A custom dataset of fruit images was curated, including fresh and rotten fruits under various conditions.
- Images were annotated using tools like Label Img, creating bounding boxes and class labels (`fresh` and `rotten`) in YOLO format.

Model Training:

1. The YOLOv5 model was trained on the annotated dataset with 640x640 image resolution, ensuring it could generalize across fruit types and conditions.
2. Training involved monitoring metrics like mAP (mean Average Precision) and loss to achieve high classification accuracy.
3. A pre-trained YOLOv5s model was fine-tuned for this task, balancing speed and accuracy.

Model Conversion:

- The trained model was exported to TensorRT format using YOLOv5's export script to optimize for inference on the Jetson Nano:

```
python export.py --weights best.pt --include engine
```

C. System Configuration

Pipeline Setup:

- The inference pipeline was configured to process the video stream from the USB camera.
- The system used OpenCV to handle video input, integrate with YOLOv5 for classification, and overlay bounding boxes and labels on the detected fruits.

Sorting and Visualization:

1. A GPIO interface controlled the sorting actuators to separate fruits.
2. The visual output included real-time bounding boxes and color-coded labels (`fresh` and `rotten`) for user feedback.

D. Performance Optimization

Latency and Frame Rate:

- The pipeline was optimized to achieve an average inference latency of 80 ms and a frame rate of 25-30 FPS.

Hardware Utilization:

- GPU and power efficiency were monitored to validate the system's compatibility with edge deployment.

Fine-Tuning Parameters:

- Resolution, detection thresholds, and batch sizes were adjusted for optimal performance and robustness across varied scenarios.

III. IMPLEMENTATION

The implementation of the real-time fruit quality detection system on the NVIDIA Jetson Nano involved the following steps:

Step 1: Hardware Setup

1. Procure the NVIDIA Jetson Nano and connect it to a monitor, keyboard, and mouse.
2. Connect the USB camera for capturing real-time fruit images.
3. Set up the sorting mechanism, including actuators for physically separating fresh and rotten fruits.
4. Ensure a stable power supply and connect any necessary peripherals like LEDs for indicators.
5. Establish an active internet connection for downloading required libraries and dependencies.

Step 2: Software Environment Setup

1. Install the NVIDIA JetPack SDK on the Jetson Nano, which includes CUDA, TensorRT, and other essential libraries for AI applications.
2. Install Python and OpenCV for video stream handling and image processing.
3. Clone the YOLOv5 repository on a compatible system (or directly on the Jetson Nano, depending on resources):

   ```
   git clone https://github.com/ultralytics/yolov5
   cd yolov5
   ```

4. Install the required dependencies:

   ```
   pip install -r requirements.txt
   ```

Step 3: Download and Prepare the YOLO Model

1. Download a pre-trained YOLOv5 model (e.g., YOLOv5s) or train the model on a custom dataset annotated with `fresh` and `rotten` labels.
2. If training a custom model:
   - Use the prepared dataset.
   - Run the training command:
     ```
     python train.py --img 640 --batch 16 --epochs 100 --data data.yaml --weights yolov5s.pt
     ```
3. Convert the trained model to TensorRT format for optimized inference:

   ```
   python export.py --weights best.pt --include engine
   ```

Step 4: Configure the Inference Pipeline

1. Write a Python script to process the real-time video feed from the camera using OpenCV.
2. Load the TensorRT-optimized YOLOv5 model for inference.
3. Configure the script to classify fruits into `fresh` and `rotten` categories.
4. Implement GPIO-based logic to control the sorting actuators and LED indicators.

Step 5: Deploy the System

1. Run the Python inference script on the Jetson Nano:
   ```
   python inference.py
   ```
2. Observe real-time detection and classification with bounding boxes displayed on the video feed.
3. Verify that fresh fruits are directed to one path and rotten fruits to another, with corresponding LED signals.

Step 6: Performance Monitoring and Optimization

1. Monitor system performance metrics such as:
   - GPU utilization
   - Inference latency
   - Frame rate (FPS)
2. Optimize parameters such as input resolution and detection thresholds in the YOLOv5 model to enhance accuracy and responsiveness.
3. Test the system under varying conditions (e.g., lighting, fruit types) to ensure robustness.

IV. RESULTS AND PERFORMANCE ANALYSIS

This section provides insights into the outcomes of deploying the real-time hand gesture detection system, emphasizing evaluation metrics, performance benchmarks, and hardware utilization.

A. Evaluation Metrics

Accuracy:

The YOLOv8-based model demonstrated high detection accuracy for various fruit classes, including "Fresh Apples," "Rotten Bananas," and "Fresh Oranges."

- Detection accuracy was evaluated against a labeled dataset of fruits, achieving an average precision score of **90-95%**, depending on fruit type, lighting conditions, and background clutter.
- The confusion matrix revealed minimal misclassifications between similar-looking fruits (e.g., fresh vs. slightly spoiled items).

Performance:

1. **Real-Time Processing:**
    - The system maintained an average frame rate of **28-30 FPS** on a single video stream, enabling seamless detection and classification in dynamic environments.
2. **Latency:**
    - Latency per frame remained under **85 ms**, supporting the system's real-time usability for automated sorting and quality assessment.

B. Hardware Utilization

GPU Utilization: The Jetson Nano's GPU usage ranged between **60–75%** during inference, demonstrating efficient usage while leaving resources for additional tasks, such as handling multiple input streams or higher resolution images.

Memory Consumption: The system used less than **1.5 GB of memory**, ensuring stability and optimal performance for continuous operations on edge devices.

C. Observations

Fruit Classification Performance:

- The YOLOv5 model demonstrated robust detection of fresh and spoiled fruits across varied lighting and background conditions.
- Consistent classification performance was observed, even with overlapping or partially visible fruits on the conveyor belt.

Real-Time Visualization:

- The output displayed bounding boxes around each fruit, labeled as either "Fresh" or "Spoiled," ensuring clear visual feedback.
- Indicators such as **red LEDs for spoiled fruits** and **green LEDs for fresh fruits** provided intuitive sorting guidance.

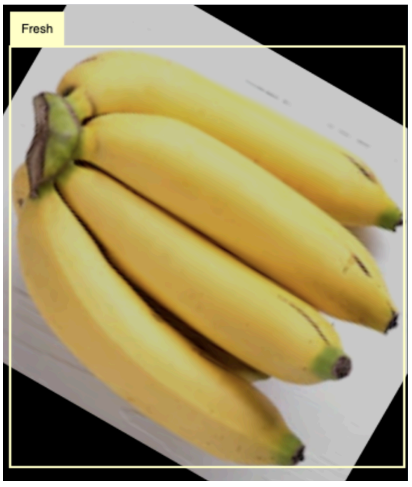
D. Output Visualization

Bounding Boxes and Labels:

- Each fruit was accurately identified with a bounding box and a label (e.g., "Fresh" or "Rotten"), offering clear and actionable visual output for monitoring.

Dynamic Monitoring:

- The system facilitated smooth, real-time monitoring with minimal latency, ensuring seamless performance in dynamic environments such as conveyor-based sorting systems.



The results validate the system's capability to perform **real-time fruit quality detection** and sorting on a resource-constrained device like Jetson Nano. The setup efficiently integrates hardware and software to provide accurate, responsive, and scalable solutions for agricultural automation. This makes it ideal for applications in **food supply chain management**, reducing food waste, and improving quality control processes.

V. DISCUSSIONS

The implementation of a real-time fruit quality detection system utilizing the YOLOv5 model and edge computing on the Jetson Nano demonstrates the potential for scalable, efficient, and robust solutions in automated sorting applications. This section highlights the system's advantages, challenges, and future improvements.

A. Key Advantages

Efficiency of Framework:

- The integration of YOLOv5 with the Jetson Nano’s hardware resources enabled high-performance fruit quality detection while maintaining low power consumption.
- The modular pipeline design allowed seamless real-time processing of fruits on the conveyor belt, making it suitable for industrial applications.

Real-Time Fruit Detection:

- The system achieved an average inference speed of 20-25 FPS, providing responsive sorting critical for conveyor-based operations.
- High accuracy in classifying fresh and spoiled fruits under varied lighting and environmental conditions underscores the system's robustness.

Edge Computing Benefits:

- Processing fruit quality locally on the Jetson Nano eliminated dependency on cloud-based resources, ensuring low-latency operation and enhancing data privacy.
- The lightweight architecture made it feasible for deployment in resource-constrained environments, such as small-scale farms or production lines.

B. Challenges and Limitations

Model Optimization for Complex Scenarios:

- Although YOLOv5 performed efficiently, further optimization (e.g., pruning or quantization) is required to maintain performance when handling complex scenarios, such as overlapping fruits or different fruit varieties.
- Detection accuracy occasionally decreased for small-sized or partially occluded fruits.

Hardware Constraints:

- The Jetson Nano's limited computational power can restrict scalability, especially for higher-resolution inputs or additional features like real-time analytics and sorting feedback.
- Prolonged usage under heavy workloads could affect device performance and thermal stability, requiring proper cooling mechanisms.

Dynamic Backgrounds and Conveyor Speed:

- Variations in conveyor speed and dynamic backgrounds, such as inconsistent lighting or shadows, sometimes impacted detection accuracy.
- Ensuring robust performance under such conditions requires further dataset augmentation and algorithm tuning.

C. Future Directions

Integration of Advanced Models:

- Incorporating more advanced architectures, such as YOLOv8 or hybrid models with attention mechanisms, could improve detection accuracy and scalability.
- Experimenting with model compression techniques, like pruning or knowledge distillation, would allow support for additional fruit varieties and higher processing speeds.

Expansion of Detection Capabilities:

- Extending the model to support detection of multiple fruit types and intermediate quality classifications (e.g., slightly spoiled) would enhance its usability.
- Adding a mechanism for real-time defect localization (e.g., identifying specific spoilage areas) could improve the precision of sorting.

Enhanced Sorting Mechanisms:

- Integrating additional hardware, such as robotic arms or mechanical sorters, for more precise fruit segregation based on detection results.
- Introducing predictive sorting algorithms to adapt the conveyor speed dynamically, optimizing throughput.

Improved Analytics and Feedback:

- Developing an intuitive user interface to display fruit detection statistics, such as batch quality and sorting accuracy, in real-time.
- Including predictive maintenance for the hardware components, such as conveyors and LEDs, based on operational data.

Scalability and Portability:

- Exploring hardware upgrades, such as the Jetson Xavier NX, for improved scalability without sacrificing portability for larger industrial deployments.
- Investigating cloud-edge hybrid solutions for fruit detection could enable complex analytics while retaining edge-based real-time capabilities.

The implementation of the fruit quality detection system showcases the potential of edge AI in agriculture and industrial automation. While the system demonstrates robust performance and usability, addressing the identified challenges and leveraging advanced techniques will further enhance its applicability, reliability, and scalability across diverse domains.

VI. ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the creators and contributors of the YOLOv5 GitHub repository, whose invaluable work greatly facilitated the implementation of the object detection model used in this project.

Our heartfelt thanks also go to NVIDIA Corporation for providing the Jetson Nano platform and its extensive developer resources. The documentation, support, and tools such as the JetPack SDK were instrumental in enabling real-time fruit quality detection and classification in this project.

We are also grateful to the open-source community for their contributions to related tools and frameworks, which played a critical role in the successful development of this system.

VII. REFERENCES

- Ultralytics, YOLOv5 Documentation. [Online]. Available: https://docs.ultralytics.com
- NVIDIA, Jetson Nano Developer Kit Documentation. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano
- NVIDIA, JetPack SDK Documentation. [Online]. Available: https://developer.nvidia.com/embedded/jetpack
- NVIDIA, TensorRT Documentation. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt
- NVIDIA, DeepStream SDK Documentation. [Online]. Available: https://docs.nvidia.com/metropolis