Fruit Decay Detection System Deployment Using CNN and YOLO Techniques

*SCM REPORT

1st S PAVITHRA
Integrated Mtech Software Engineering
VIT CHENNAI
Chennai,India
pavithra.s2022@vitstudent.ac.in

2nd T MAHENDRA BABU
Integrated Mtech Software Enginnering)
VIT CHENNAI
Chennai,India
mahendrababu.t2022@vitstudent.ac.in

3rd SATHVIK GUPTHA
Integrated Mtech Software Engineering
VIT CHENNAI
Chennai,India
sathvikguptha.a2022@vitstudent.ac.in

4th GURUVELLI TARUN
Integrated Mtech Software Engineering
VIT CHENNAI
Chennai,India
guruvelli.tarun2022@vitstudent.ac.in

Software, especially AI-based systems like Fruit Decay Detection, evolves over time. The system must adapt to changing market needs, new fruit varieties, updated decay patterns, and technological advancements. By integrating continual maintenance throughout the product lifecycle, we can ensure the system's sustainability and relevance in the field.

Current Version: Version 1.0 - Basic functionality to upload images, detect fruit, and classify freshness or decay.

***Index Terms*—**

I. Requirements Analysis

Before deployment, an extensive requirements analysis is conducted by: User Interviews: Understanding the needs of end-users such as farmers, distributors, or warehouse operators. Studying Existing Systems: Exploring existing methods for decay detection or alternatives, identifying gaps that CNN and YOLO models can fill. Building a Prototype: Developing an initial version using OpenCV for image preprocessing and testing detection using pre-trained YOLO models. Feedback from users, field experts, and earlier versions of similar products help refine the system

II. Design Phase

The design phase follows key software engineering principles: Standards Compliance: Ensuring compatibility with Amazon Sagemaker's APIs, OpenCV libraries, and any external systems for data input (e.g., cameras). Portability: Using Docker containers to make the system portable across different cloud environments, from AWS to edge devices. User Interface: Designing an intuitive API or web-based interface for users to upload images and view decay predictions. Maintainability: Modular design of the code, allowing independent updates for the CNN model, YOLO configuration, or OpenCV pipelines without affecting the entire system. Interoperability: Ensuring seamless integration with third-party systems such as agricultural IoT platforms.

Identify applicable funding agency here. If none, delete this.

III. Development Phase

During the development, the following practices ensure high-quality, maintainable code: Proper Documentation: Comprehensive documentation of datasets, model architecture, training process, and deployment configurations. Code Reusability: Implement reusable modules, such as pre-built pipelines for image augmentation (using OpenCV) and data handling (Amazon Sagemaker SDK). Adherence to Coding Standards: Ensuring consistent coding practices across different components (e.g., Python code for YOLO, OpenCV image handling, Sagemaker deployment scripts). SQA Procedures: Software Quality Assurance (SQA) is implemented through peer code reviews, automated unit tests, and performance checks

IV. Testing Phase

Thorough testing of the system ensures its robustness before deployment: White Box Testing: Internal testing of the CNN and YOLO models, ensuring each layer works as expected. Black Box Testing: Evaluating the system as a whole, testing its ability to detect decay in various real-world conditions (e.g., poor lighting, occluded fruits). Integration Testing: Ensuring smooth communication between the CNN, YOLO, OpenCV for preprocessing, and Sagemaker for inference. System Testing: Testing the entire system end-to-end with real-time data

feeds and edge cases. Regression Testing: Regularly testing the system with previous versions of the software to ensure that updates don't introduce bugs

V. Deployment Models

The system is expected to be deployed in the following strategies: Cloud-Based Deployment: Leveraging Amazon Sagemaker to manage model training, deployment, and scaling. Sagemaker endpoints allow for real-time decay detection as images are uploaded. Edge Deployment: For environments with limited internet access (e.g., farms), the system can be deployed on edge devices using pre-trained YOLO models and OpenCV for local image processing.

VI. Maintenance Phase

1. Operational Maintenance Ensures the system has the resources and infrastructure to function smoothly. Resource Monitoring: Track usage of cloud (Sagemaker, storage) and edge devices, ensuring enough resources are available for processing. Backups: Regularly back up models and datasets to prevent data loss. Performance Tuning: Optimize system performance, scaling resources based on demand (especially on AWS). 2. Corrective Maintenance Fixes issues or bugs that may arise during system operation. Bug Fixing: Address false positives or negatives in decay detection. API Issues: Resolve issues in Amazon Sagemaker API to maintain real-time inference. Hardware Failures: On edge devices, correct camera or sensor malfunctions. 3. Preventive Maintenance Proactive measures to prevent issues before they occur. Model Evaluation: Periodically test models to ensure accuracy doesn't degrade over time. Regular Updates: Retrain models with new data to prevent them from becoming outdated. Health Checks: Regularly inspect cloud and edge systems, updating software as necessary. 4. Adaptive Maintenance Modify the system to accommodate changes in technology or user requirements. New Fruit Types: Update models to detect decay in new fruit types or conditions. Technology Upgrades: Adapt the system to work with new hardware or updated software (OpenCV, YOLO, etc.). Platform Adaptation: Make the system compatible with new devices or platforms (mobile apps, industrial cameras).

VII. Version Planning

Version 1.1 Error Reduction and Improved Usability: Eliminate edge-case errors and enhance user prompts. This will involve refining error handling to make it more user-friendly and informative. Implementation: Structured error messages and clear upload guidelines will improve user experience, guiding optimal image capture. 2. Version 1.2 Expanded Detection Capabilities: Support multi-fruit detection, allowing users to analyze multiple fruits within a single image. Implementation: Adjust YOLO and CNN to handle multiple objects and test with various fruit combinations to validate accuracy. 3. Version 2.0 - UI Enhancements and Architecture Optimization: Introduce an improved UI and optimize backend architecture for scalability. Implementation: Use responsive design principles for a more intuitive interface and configure SageMaker for scaling as user demand grows.