Building an Automated Fault Detection System for Production Lines Using Raspberry Pi and Machine Learning

Introduction

This project provides a comprehensive guide to creating an automated fault detection system for production lines using Raspberry Pi and Machine Learning. The system is designed to capture, train, and detect flaws in products, offering a cost-effective solution for automating quality control processes.

Overview of Steps

Step 1: Hardware Set Up

- Raspberry Pi Setup:
  - Insert a MicroSD card with Raspberry Pi OS.
  - Connect the Raspberry Pi to a power supply and attach the camera module.
- Ultrasonic Sensor Installation:
  - Mount the ultrasonic sensor before the camera to detect passing products.
  - Configure the sensor to trigger the camera for image capture.
  - Establish communication between the Arduino (if used) and Raspberry Pi using I2C, serial, or GPIO.
- Camera and Conveyor Belt Setup:
  - Position the camera to capture clear images of products.
  - Optional: Use a stepper motor to control a conveyor belt, managed by the Raspberry Pi or a microcontroller like Arduino.
- Lighting and Background:
  - Use consistent lighting and a uniform white background to improve image contrast and defect detection accuracy.

Step 2: Capturing Images for Training

- Install Picamera2:
  - Update and install the Picamera2 library using:

```
sudo apt update
sudo apt install python3-picamera2
```

- Image Collection:
  - Use the provided Python script (`images.py`) to capture images.
  - Organize images into two folders: “correct” (flawless products) and “incorrect” (defective products).
- Dataset Preparation:
  - Split the dataset into training and testing sets with a recommended ratio of 70/30 or 80/20.

Step 3: Setting Up the CNN Model

- Install Required Libraries:
  - Install TensorFlow, NumPy, and related libraries using:
    ```bash
    pip install tensorflow numpy Pillow
    ```
  - Perform these installations in a virtual environment for better management.
- Model Training:
  - Modify the provided Python script (`creation.py`) to:
    - Specify paths for the training and validation datasets.
    - Define a save location for the trained model.
  - Run the script to train the Convolutional Neural Network (CNN) for defect detection.
  - Save the trained model for deployment.

Step 4: Testing the Model

- Image Testing:
  - Capture new images and test the trained model using the provided script (`testing_model.py`).
- Accuracy Calculation:
  - Calculate the model's accuracy using:
    ```
    Accuracy = (Number of Correct Predictions / Total Predictions) × 100%
    ```
- Improvement:
  - If accuracy is unsatisfactory, retrain the model with more images and better lighting.

Step 5: Microcontroller Deployment

- Use HTTP communication between the Raspberry Pi and Arduino for motor control and image capture.
- The Arduino detects objects and sends a signal to the Raspberry Pi for image evaluation.
- The provided Arduino script (`project_final.ino`) manages communication and control.

Step 6: Final Deployment

- User Interface:
  - Download and run the final Python script (`final_ver.0.1.py`).
  - The script creates a GUI with a button to evaluate captured images and display accuracy.
- Automation:
  - If connected to a microcontroller, image capture and evaluation occur automatically.

Supplies Required

- Raspberry Pi 5 (or any model with a camera interface).
- Camera Module (Raspberry Pi Camera Module or any USB camera).
- Ultrasonic Sensor (to detect products).

- Conveyor Belt and Stepper Motor (optional).
- Power Supply, MicroSD card, and Lighting Setup.

Attachments

The project includes several downloadable scripts for different stages:
- Image Capture Script: `images.py`
- Model Creation Script: `creation.py`
- Model Testing Script: `testing_model.py`
- Arduino Microcontroller Script: `project_final.ino`
- Final Deployment Script: `final_ver.0.1.py`

Key Benefits of the System

- Cost-Effective: Uses affordable hardware like Raspberry Pi.
- Automated QA: Streamlines defect detection in production lines.
- Customizable: Easy to adapt for different product types and defect categories.
- Scalable: Can integrate with existing production systems.

Recommendations

This project is ideal for industries looking to implement budget-friendly automated quality control. For improved results:
- Ensure consistent lighting and a controlled environment for image capture.
- Continuously improve the model by adding diverse training data.

Conclusion

I provide a practical and detailed approach to building an automated fault detection system. Combining Raspberry Pi with machine learning ensures a modern solution for quality assurance, making it accessible to small and medium-sized enterprises.