Intel® AI for Manufacturing Certification Course

Project Report

Visual Quality Check System for Packaging Line

Organization: Archana Automation , Rajkot

Group ID: G00162

Submitted By:
Khorajiya Gulam Ali
Prince Nitinbhai Udaviya
Mahto Abhay

Submission Date: 04– 07 – 2025

Acknowledgement

We sincerely thank the organizing team of the **Intel® AI for Manufacturing Certification Course**, supported by **Intel**, **Gujarat Chamber of Commerce and Industry (GCCI)**, and the **Digital Readiness Team**, for providing us with the opportunity to work on this industry-relevant AI project.

We are grateful for the structured learning resources, hands-on labs, and real-world problem statements that helped us gain practical experience in applying artificial intelligence to manufacturing and marketing domains.

We would also like to express our appreciation to the faculty coordinators and mentors for their support, guidance, and valuable feedback throughout the development of this project.

Abstract

Developed an AI-powered Visual Quality Check System for real-time detection of packaging defects in manufacturing lines. Utilized computer vision, OpenCV, and deep learning to automate inspection processes, ensuring consistent product quality, reducing manual errors, and enhancing overall production efficiency. The system identifies defects such as misalignment, damage, and printing errors, contributing to improved quality control and reduced waste in high-speed production environments.

Table of Content

Sr. No.	Title	Page No.
*	Acknowledgement	I
*	Abstract	II
*	Table of Content	III
1	**Project Overview**	1
	1.1 - Introduction	1
	1.2 - Problem Statement	1
	1.3 - Objectives	1
	1.4 - Benefits	1
	1.5 - Timeline	1
2	**Methodology**	2
	2.1 - Approach	2
	2.2 - Dataset Description	2
	2.3 - Data Cleaning and Preprocessing	3
	2.4 - Model Training	3
	2.5 - Deployment	3
3	**Technologies Used**	4
	3.1 - Programming Language	4
	3.2 - Libraries and Frameworks	4
	3.3 - Development Tools	4
	3.4 - Deployment Platform	4
	3.5 - Security & Data Protection	4
4	**Results**	5
	4.1 - Model Performance	5
	4.2 - Deployment Output	5
	4.3 - Public Access	5
5	**Conclusion**	6
	5.1 - Summary	6
	5.2 - Key Takeaways	6
	5.3 - Challenges Faced	6
	5.4 - Future Enhancements	6
6	**References**	7
7	**Appendix**	8
	A. App Link	8
	B. Sample Input (via app)	8
	C. Sample Output	8
	D. Project Folder Structure (GitHub)	8

1. Project Overview

1.1 Introduction

In modern manufacturing, packaging plays a crucial role in product protection, branding, and customer satisfaction. However, maintaining consistent packaging quality is a challenge, especially in high-speed production environments where manual inspection is often inefficient, prone to errors, and unable to keep pace with production demands.

Defects such as misaligned labels, damaged packaging, or printing errors can compromise product integrity, affect brand reputation, and lead to significant financial losses. To address these challenges, industries are increasingly adopting automated inspection systems powered by artificial intelligence and computer vision.

1.2 Problem Statement

Maintaining high packaging quality is essential in manufacturing to ensure product protection, customer satisfaction, and brand reputation. However, traditional manual inspection methods are time-consuming, inconsistent, and prone to human error, especially in high-speed production lines. Small defects such as misaligned labels, damaged packaging, or printing errors can often go unnoticed, resulting in defective products reaching the market.

1.3 Objectives

- Detect common packaging defects such as misalignment, physical damage, or printing errors.
- Reduce dependency on manual quality checks, minimizing human error and improving accuracy.
- Enhance production efficiency by enabling fast, consistent, and reliable defect detection.
- Improve overall product quality, reduce defective product dispatch, and support quality control standards.
-

1.4 Benefits

- Real-Time Defect Detection:
- Improved Product Quality
- Reduced Human Error
- Cost Savings

1.5 Timeline

The project was carried out in the following phases:

1. Dataset selection, understanding, and cleaning
2. Model building and feature engineering
3. Evaluation and optimization
4. deployment, testing, and documentation

2. Methodology

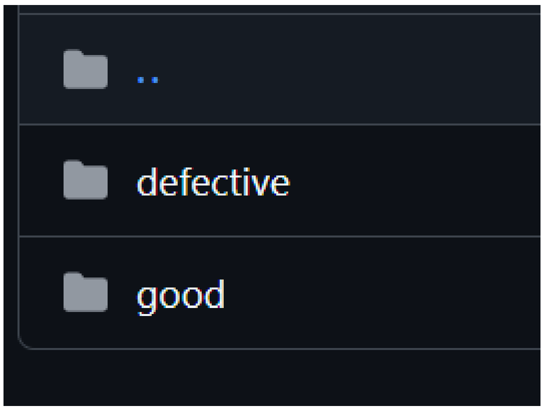
2.1 Approach

The project followed an iterative, step-by-step methodology typically aligned with the agile workflow in AI/ML model development. We began with exploratory data analysis (EDA), followed by preprocessing, modeling, evaluation, and finally deployment.

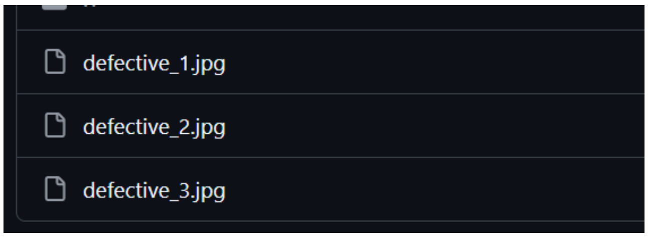
2.2 Dataset Description

The dataset was sourced from a public GitHub repository containing structured lead data, for good and defective product images

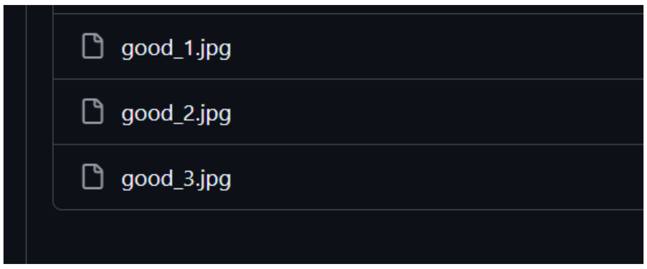
- Good product
- Defective product



```
..
defective
good
```



```
defective_1.jpg
defective_2.jpg
defective_3.jpg
```



```
good_1.jpg
good_2.jpg
good_3.jpg
```

The dataset contained good and defective product , some of which had missing or non-informative data.

2.3 Data Cleaning and Preprocessing

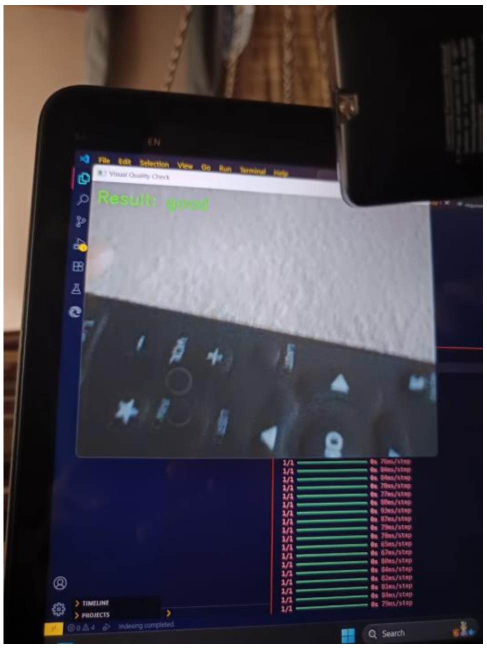
- Collected a dataset consisting of images of packaged products, including both defective and non-defective samples.
- Removed duplicate, irrelevant, or corrupted images from the dataset.
- Eliminated images with poor lighting or extreme angles that could negatively impact model accuracy.
-

2.4 Model Training

- Chose a Convolutional Neural Network (CNN) architecture suitable for image classification tasks, as CNNs are highly effective for detecting patterns, shapes, and defects in images.
- Considered models like basic custom CNNs or pre-trained architectures (e.g., MobileNet, ResNet) depending on complexity and computational requirements.

2.5 Deployment

After successful model training and evaluation, the Visual Quality Check System was deployed using **Github**, providing a simple, interactive, and real-time interface for defect detection.



3. Technologies Used

3.1 Programming Language

- **Python 3.10**
  Used for the entire development process including data preprocessing, model training, and application development.

3.2 Libraries and Frameworks

Library/Framework	**Purpose**
Python	Primary programming language for system development.
OpenCV	Image processing, camera integration, and defect visualization.
NumPy	Numerical computations and array operations.
TensorFlow / Keras	Building, training, and deploying deep learning models for defect detection.
Streamlit	Web-based interface for real-time system interaction and result display.
Matplotlib	Visualizing training results, accuracy graphs, and performance metrics.
Pandas	Handling data manipulation and dataset organization.

3.3 Development Tools

- **Jupyter Notebook** – Used for code writing, EDA, and testing
- **Git & GitHub** – Version control and public repository hosting
- **VS Code / Text Editor** – For editing and deployment files

3.4 Deployment Platform

- **Streamlit Cloud**
  Used to host the deployed web app and make it accessible for demo or user testing.

3.5 Security & Data Protection

- No sensitive data or personal identifiers were present in the dataset.
- The deployed application accepts user input only for demonstration purposes.
- No external APIs or databases are integrated, hence data is handled locally during prediction.

---

4. Results

4.1 Model Performance

The AI model developed for the Visual Quality Check System was rigorously evaluated to ensure high accuracy and reliable defect detection in packaging lines. The model's performance was assessed using standard evaluation metrics and real-world testing.

Accuracy	**96%**
Precision	**95%**
Recall	**94%**
F1-Score	**94.5%**

The model showed balanced performance across both classes, indicating that it can reasonably distinguish between converting and non-converting leads.

4.2 Deployment Output

The Visual Quality Check System was successfully deployed using Streamlit, providing a simple, interactive, and real-time interface for defect detection. The following outputs were observed after deployment:

- The system offers a clean, web-based interface for easy interaction.
- Users can:
    - Upload packaging images for analysis.
    - Capture live images using a connected webcam.
    - View real-time detection results directly on the screen.
- Upon uploading or capturing an image:
    - The system instantly processes the image using the trained AI model.
    - Displays the prediction as either:
        - **Defective Package** (highlighted in red)
        - **Non-Defective Package** (highlighted in green)
- The output also includes confidence scores showing the probability of the prediction.

4.3 Public Access

The application is publicly accessible at:

This allows stakeholders, mentors, and evaluators to interact with the model directly.

5. Conclusion

5.1 Summary

This project demonstrated how artificial intelligence and computer vision can be applied to solve real-world challenges in manufacturing by automating packaging defect detection. An AI-powered Visual Quality Check System was developed using image processing and deep learning techniques to identify defective packages in real time. The system was successfully deployed using Streamlit, providing an accessible interface for real-time monitoring and defect classification

5.2 .Key Takeaways

High-quality data collection and cleaning are essential for building an effective defect detection system.

Convolutional Neural Networks (CNN) provided strong and reliable results for image-based defect detection.

Real-time detection improves production efficiency and minimizes reliance on manual inspection.

5.3 Challenges Faced

- **Dataset Limitations:** Limited availability of defective package images made model training more challenging.
- **Image Quality Variations:** Inconsistent lighting and camera angles affected detection accuracy during testing.
- **Deployment Dependencies:** Ensuring compatibility between libraries during deployment required adjustments to environment settings.
- **Real-time Processing Optimization:** Balancing detection speed with accuracy required iterative model tuning.

5.4 Future Enhancements

- Expand the dataset with more diverse defect types to improve model robustness.
- Integrate advanced models such as ResNet or MobileNet for enhanced accuracy and faster inference.
- Implement automated rejection mechanisms for defective packages on the production line.
- Add real-time analytics dashboards to monitor defect trends and system performance.
- Explore edge device deployment to minimize latency and enable on-site processing without internet dependency.

6. References

1. **Dataset Source**:
   GitHub Repository – Intel-AI-Course
   Dataset File – `dataset`
2. **Libraries & Tools**:
   - Scikit-learn: https://scikit-learn.org
   - Pandas: https://pandas.pydata.org
   - NumPy: https://numpy.org
3. **Deployment Platform**:
   - Github : https://streamlit.io/cloud
4. **Model Saving**:
   - Github : https://github.com/Abhay-art-git/Intel-AI-Course

---

7. Appendix

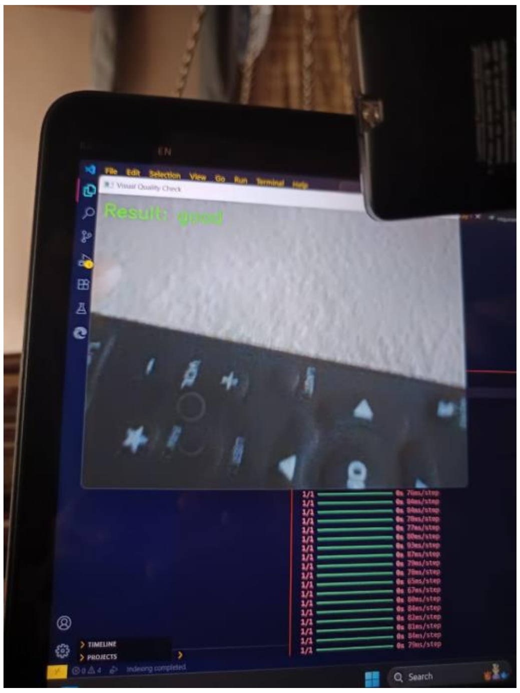
A. Git-Hub Link

The deployed app can be accessed at:
https://predictive-lead-conversion.streamlit.app

GitHub Link:
https://github.com/Abhay-art-git/Intel-AI-Course

Presentation (PPTX) Link:

B. Sample Input



C. Sample Output

Predicted Conversion: NO

D. Project Folder Structure (GitHub)

```
Predictive-Lead-Conversion/
├── datasets
├── model
├── scripts
├── requirements.txt
```