INTEL-UNNATI AI&ML JOURNAL
Number 2 (2025), pp. 1–17 Code:IUIT-2025

MACHINE LEARNING

Smart Product Labeling and Traceability System for Quality Control Automation

Anna Boban, Nivya Mathew, and Noopa Maria Rojan

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: In today's fast-moving manufacturing industry, keeping track of product quality and origin has become more important than ever. This project presents a Smart Labeling and Traceability System that aims to make that process more reliable, accurate, and automated. The system is designed to identify each product, check basic quality parameters, and apply or verify key details such as Device ID, Batch ID, Manufacturing Date, RoHS compliance, and a unique Serial Number in the form of a QR code or barcode. Instead of relying on people to inspect each item, this system uses a mix of sensors, actuators, and AI to automatically identify products, verify important details such as device ID, batch number, manufacturing date, and RoHS compliance, and apply or confirm labels with QR codes. It does not stop there; the system also keeps a full digital record of every product it processes, so manufacturers can trace anything back to its source if needed. That kind of visibility is a big deal in today's quality-driven environment. We built and tested the whole setup using tools such as Tinkercad, Google Colab, and Proteus, which made the development process both practical and educational. The result is a reliable, forward-thinking solution that reflects the spirit of Industry 4.0, where automation and intelligence work hand in hand to improve how things are made.

1 Introduction

In today's fast-paced electronics industry, ensuring product quality, regulatory compliance, and traceability has become more important than ever. With increasing global focus on environmental standards like RoHS (Restriction of Hazardous Substances), manufacturers are under pressure to not only deliver functional products but also prove that those products meet strict safety and environmental requirements.

This project presents a smart and automated solution to address that challenge.

We have developed an intelligent real-time product traceability and labeling system that leverages AI-based object detection, RoHS compliance verification, and QR code labeling to create a complete digital audit trail for each product. The system uses a trained YOLOv8 model to identify products like remotes, hard disks, and lamps from a live camera feed. Once identified, the system cross-references each product with detailed RoHS metadata stored in an Excel file, verifies its compliance status, and then generates a unique label containing traceability information and a QR code.

Every label includes key manufacturing details such as the product model, serial number, batch ID, operator ID, timestamp, and RoHS status. This data is also logged in a CSV file for traceability, enabling quality checks, audits, and accountability.

The system is designed to be lightweight, modular, and suitable for automated production lines where compliance and transparency are essential. By combining computer vision, data automation, and digital labeling, this solution demonstrates how AI can be practically used to streamline manufacturing compliance and reduce human error while keeping the process fast, reliable, and scalable.

2 Libraries Used

In the project for various tasks, following packages are used.

```
    Ultralytics
    OpenCV (cv2)
    qrcode
    Pillow (PIL)
    Pandas
    datetime
    os
```

3 Methodology

This project focuses on the development of an AI-driven traceability and compliance verification system for electronic products. The methodology integrates real-time object detection using a trained deep learning model, RoHS (Restriction of Hazardous Substances) compliance verification using regulatory metadata, and automated label generation with QR encoding for product traceability.

1. **Product Metadata Definition :** A product metadata structure was developed to store key attributes for each item in the production line. This included the product model, components used, supplier names, voltage requirements, warranty period, and RoHS compliance status. This metadata served as the foundational reference for each product's identity and regulatory profile.

2. **RoHS Compliance Verification from Excel :** To ensure regulatory conformity, RoHS compliance data was maintained in an external Excel spreadsheet. The sheet detailed individual substance concentrations for each product, including lead (Pb), cadmium

(Cd), mercury (Hg), hexavalent chlorium (Cr6 +), polybrominated biphenyls (PBB) and polybrominated diphenyl ethers (PBDE). Each product's compliance status ("Yes" or "No") was paired with an explanatory message that highlighted the reason for compliance or non-compliance. These data were dynamically loaded into the system and served as a source to verify the RoHS status of each detected product.

3. **YOLOv8-Based Real-Time Detection :** A pre-trained YOLOv8 (You Only Look Once, version 8) model was used for object detection in a live video stream. The model was trained to recognize specific electronic products such as remote controls, routers, hard disks, and lamps. Upon capturing live input from a connected webcam, the system continuously processed video frames to detect and classify products in real time.

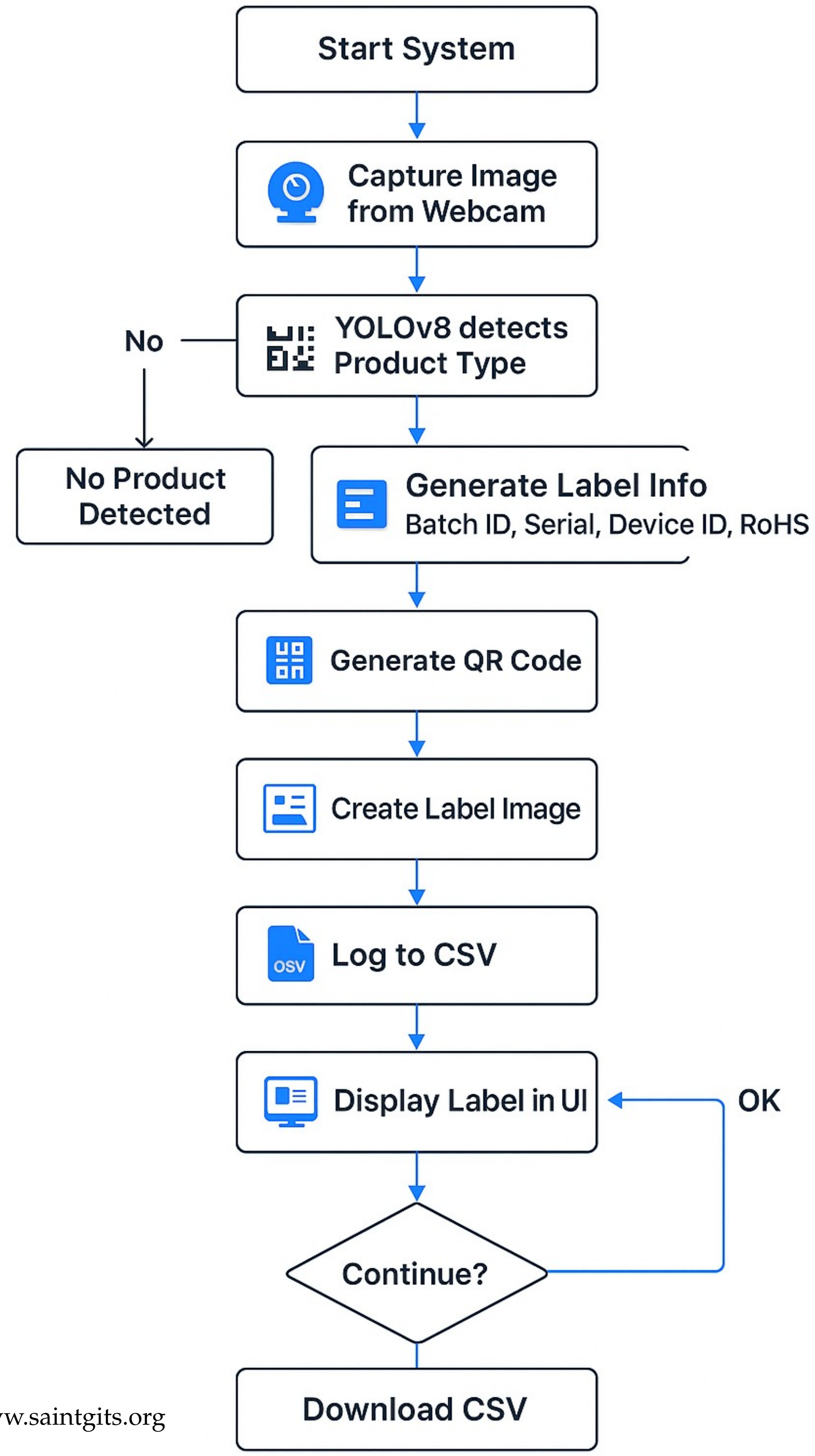
4. **Label Generation and Trace Logging :** Once a product was detected and verified, the system generated a complete digital label that included:
•Product name and model
•Batch ID and serial number
•Device ID and timestamp
•Shift and operator ID
•RoHS compliance status and explanation

A QR code containing the full traceability record was also generated and embedded into the visual label. Both the label and QR code were saved as images for printing or archival. All product details, including compliance results, were simultaneously logged into a structured CSV file to maintain a digital traceability log for compliance auditing.

5. **Batch Management and Automation :** The system was designed to operate within defined production batch limits. For each batch, a maximum number of items was specified, beyond which the system would automatically halt further labeling to avoid overflow. Each labeled item was tracked against this batch count, ensuring organized and error-free labeling per production cycle.

6. **Live Monitoring and User Interaction :** Detected products were displayed on a live video feed with bounding boxes and product names overlaid for operator confirmation. A simple keyboard-based control interface allowed operators to monitor detection status and exit the system safely upon completion.

Dataflow Architecture



4 Implementation

This project presents a complete and intelligent traceability and labeling system designed specifically for electronic product lines. The system integrates real-time object detection, compliance verification, and automatic label generation to ensure that every unit is identified accurately, checked for RoHS compliance, and logged for traceability. From capturing product data using a webcam to generating QR-coded labels and maintaining detailed CSV records, the solution streamlines quality control and enhances accountability across the production line.

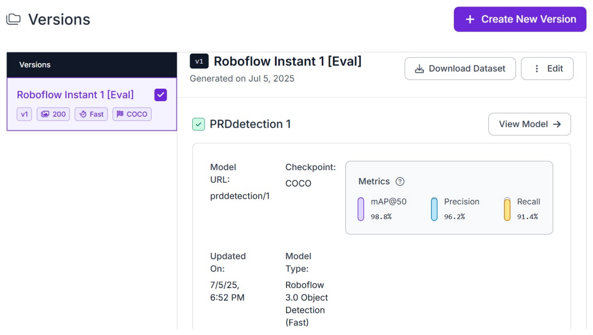
The system was built to be user-friendly and modular, allowing operators with minimal technical expertise to run it efficiently. It mimics a real-world factory scenario where products move through an inspection station, get verified, labeled, and trackedâĂŤall in real time.

1. **Development Environment :** The project was developed entirely in Google Colab, which provides a flexible Python-based environment with GPU support. This allowed us to train models, simulate industrial processes, and interact with image data in real time. The following tools were used:
•Python 3.10 as the core programming language.
•Ultralytics YOLOv8 for real-time object detection.
•OpenCV to capture webcam images.
•Pandas for reading and filtering Excel and CSV data.
•Pillow (PIL) for dynamically generating product labels as images.
•QRcode for encoding traceability information in a scannable format.
•Built-in Python modules like datetime, random, and os were used for timestamps, serial generation, and file management.

The entire system runs on a standard laptop with a webcam, but it can be easily extended with label printers or barcode scanners in industrial settings.

2. **YOLOv8 Model Training and Loading :** A robust object detection system is the foundation of our project. We used YOLOv8n, a lightweight version of YOLOv8, to keep the model fast and accurate. Here's how we trained it:
•Data Collection Annotation: Images of three products:Remote, Hard Disk, and Lamp were collected in diverse backgrounds and lighting conditions to simulate real-world factory settings.
•Annotation: Each image was annotated using tools like Roboflow to mark the object locations and assign class labels.
•Model Training: We trained the YOLOv8n model using Colab with the annotated dataset and configured it to run for 30 epochs. Training was done using the YOLOv8 API, and metrics were monitored to ensure convergence.
•Model Integration: Once the model achieved good accuracy, we exported the best.pt weights file and loaded it into our labeling system. This enabled real-time inference within the Colab environment.

This trained model allowed us to recognize each product category with high precision during live webcam capture.



Roboflow Instant 1 [Eval]
v1 200 Fast COCO

v1 Roboflow Instant 1 [Eval]
Generated on Jul 5, 2025

PRDdetection 1

Model URL: prddetection/1
Checkpoint: COCO
Metrics
mAP@50 98.8%
Precision 96.2%
Recall 91.4%
Updated On: 7/5/25, 6:52 PM
Model Type: Roboflow 3.0 Object Detection (Fast)

Figure 1: *Roboflow Dashboard Showing PRDdetecion Model Metrics Summarizes mAP50 (98.8%), precision (96.2%), and recall (91.4%) for the trained object detection model using a 200-image COCO dataset.*

3. **Product Metadata Dictionary :** For each product class, we manually defined a rich metadata dictionary that includes:

Model number
Warranty duration
Manufacturing location
Electrical/Storage specifications (e.g., voltage for LAMP, storage size for HDD)
Packing method (e.g., bubble wrap, foam box)
List of components (like microcontroller, IR LED, PCB, etc.)
Associated suppliers for each component
This metadata served multiple roles:
It populated the visual label generated for each product.
It helped simulate a detailed bill of materials and supply chain mapping.
It ensured standardization across every detected instance of a product.

Rather than hardcoding attributes every time a product is detected, we dynamically merge this metadata with real-time detection results, ensuring efficient traceability and label consistency.

4. **RoHS Compliance Verification via Excel File :** To replicate real-world regulatory checks, we built an external Excel file (rohs_compliance.xlsx) containing:

Product names
Compliance decision (Yes/No)
Explanation or reason for non-compliance (optional)

The system reads this file into a Pandas DataFrame and maps product names to their RoHS status. When a product is detected:

Its name is matched (case-insensitively) with the Excel data.
If marked as Yes, the label shows "RoHS Compliant".
If marked No, the label warns with "Not Compliant".

This Excel-driven logic makes the system flexible and auditable. You can simply update the Excel file to reflect new compliance statuses without touching the code, simulating how industries store audit trails for compliance.

5. **Real-Time Detection, Labeling, and Logging :** This is the heart of the system, an intelligent loop that handles all automation:

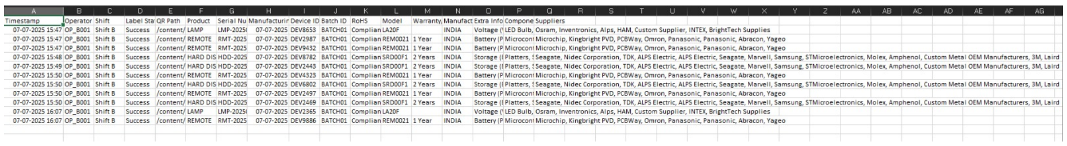
•Capture:
The webcam is triggered to take a snapshot of the current frame.
The photo is saved for inference.

•Object Detection:
The trained YOLOv8 model predicts the product class from the image.
If no product is detected, the loop notifies the operator.

•Traceability Metadata Generation:
For every product detected:
A unique serial number is created using a timestamp and a random ID.
A batch ID is computed based on how many units of that product were processed.
A Device ID is generated (simulated with random numeric IDs).
The current date, operator ID, and shift (A/B/C) are captured automatically.

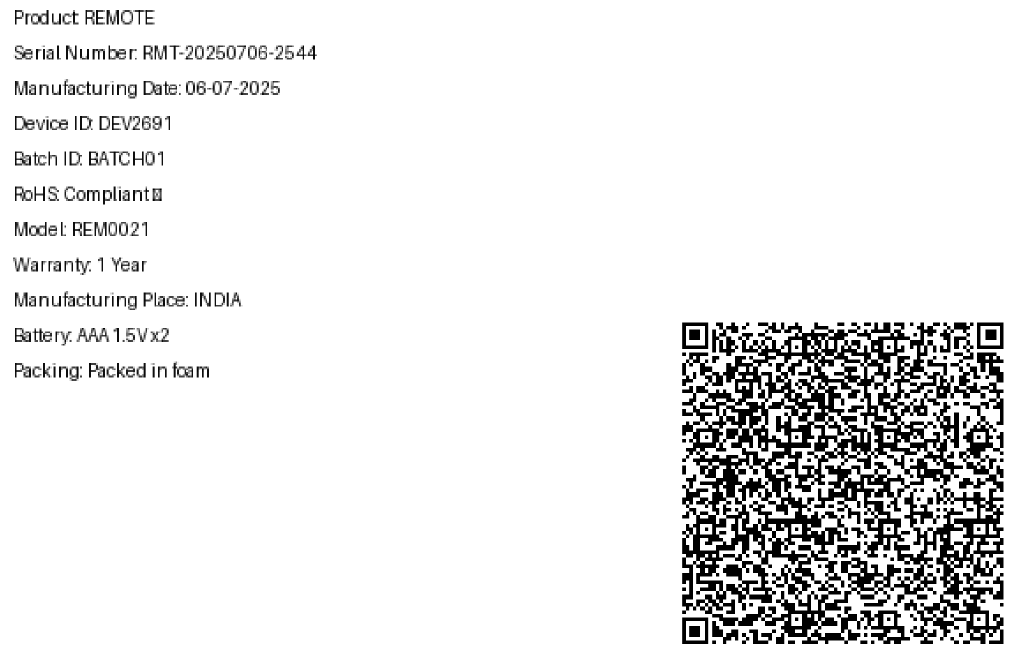
•Label and QR Code Creation:
A label is generated visually using Pillow that contains:
All the metadata and product info. A RoHS status message.
A QR code containing the full text of the product's traceability.
The QR code is saved as an image and also embedded into the visual label.

•Logging:
All traceability data is logged to a centralized CSV file (traceability_log.csv) including:
QR path
Product metadata
Compliance status
Components and supplier names
This CSV acts as the digital record for post-audit or database uploads.



Timestamp	Operator	Shift	Label Sta	QR Path	Product	Serial Nu	Manufacturir	Device ID	Batch ID	RoHS	Model	Warranty,	Manufact	Extra Info	Compone	Suppliers
07-07-2025 15:47	OP_B001	Shift B	Success	/content/	LAMP	LMP-2025	07-07-2025	DEV8653	BATCH01	Complian	LA20F		INDIA	Voltage ('	LED Bulb,	Osram, Inventronics, Alps, HAM, Custom Supplier, INTEX, BrightTech Supplies
07-07-2025 15:47	OP_B001	Shift B	Success	/content/	REMOTE	RMT-2025	07-07-2025	DEV2987	BATCH01	Complian	REM0021	1 Year	INDIA	Battery (P	Microcon	Microchip, Kingbright PVD, PCBWay, Omron, Panasonic, Panasonic, Abracon, Yageo
07-07-2025 15:47	OP_B001	Shift B	Success	/content/	REMOTE	RMT-2025	07-07-2025	DEV9432	BATCH01	Complian	REM0021	1 Year	INDIA	Battery (P	Microcon	Microchip, Kingbright PVD, PCBWay, Omron, Panasonic, Panasonic, Abracon, Yageo
07-07-2025 15:48	OP_B001	Shift B	Success	/content/	HARD DIS	HDD-2025	07-07-2025	DEV8782	BATCH01	Complian	SRD00F1	2 Years	INDIA	Storage (	Platters,	Seagate, Nidec Corporation, TDK, ALPS Electric, ALPS Electric, Seagate, Marvell, Samsung, STMicroelectronics, Molex, Amphenol, Custom Metal OEM Manufacturers, 3M, Laird
07-07-2025 15:49	OP_B001	Shift B	Success	/content/	HARD DIS	HDD-2025	07-07-2025	DEV2443	BATCH01	Complian	SRD00F1	2 Years	INDIA	Storage (	Platters,	Seagate, Nidec Corporation, TDK, ALPS Electric, ALPS Electric, Seagate, Marvell, Samsung, STMicroelectronics, Molex, Amphenol, Custom Metal OEM Manufacturers, 3M, Laird
07-07-2025 15:50	OP_B001	Shift B	Success	/content/	REMOTE	RMT-2025	07-07-2025	DEV4323	BATCH01	Complian	REM0021	1 Year	INDIA	Battery (P	Microcon	Microchip, Kingbright PVD, PCBWay, Omron, Panasonic, Panasonic, Abracon, Yageo
07-07-2025 15:50	OP_B001	Shift B	Success	/content/	HARD DIS	HDD-2025	07-07-2025	DEV6802	BATCH01	Complian	SRD00F1	2 Years	INDIA	Storage (	Platters,	Seagate, Nidec Corporation, TDK, ALPS Electric, ALPS Electric, Seagate, Marvell, Samsung, STMicroelectronics, Molex, Amphenol, Custom Metal OEM Manufacturers, 3M, Laird
07-07-2025 15:50	OP_B001	Shift B	Success	/content/	REMOTE	RMT-2025	07-07-2025	DEV2497	BATCH01	Complian	REM0021	1 Year	INDIA	Battery (P	Microcon	Microchip, Kingbright PVD, PCBWay, Omron, Panasonic, Panasonic, Abracon, Yageo
07-07-2025 15:50	OP_B001	Shift B	Success	/content/	HARD DIS	HDD-2025	07-07-2025	DEV2469	BATCH01	Complian	SRD00F1	2 Years	INDIA	Storage (	Platters,	Seagate, Nidec Corporation, TDK, ALPS Electric, ALPS Electric, Seagate, Marvell, Samsung, STMicroelectronics, Molex, Amphenol, Custom Metal OEM Manufacturers, 3M, Laird
07-07-2025 16:07	OP_B001	Shift B	Success	/content/	LAMP	LMP-2025	07-07-2025	DEV2365	BATCH01	Complian	LA20F		INDIA	Voltage ('	LED Bulb,	Osram, Inventronics, Alps, HAM, Custom Supplier, INTEX, BrightTech Supplies
07-07-2025 16:07	OP_B001	Shift B	Success	/content/	REMOTE	RMT-2025	07-07-2025	DEV9886	BATCH01	Complian	REM0021	1 Year	INDIA	Battery (P	Microcon	Microchip, Kingbright PVD, PCBWay, Omron, Panasonic, Panasonic, Abracon, Yageo

Figure 2: Automatically generated traceability log stored in an Excel sheet, capturing real-time metadata for each AI-detected product. It includes timestamp, operator details, shift, QR label status, serial number, batch ID, and component-level supply chain information for RoHS compliance.



Product: REMOTE
Serial Number: RMT-20250706-2544
Manufacturing Date: 06-07-2025
Device ID: DEV2691
Batch ID: BATCH01
RoHS: Compliant □
Model: REM0021
Warranty: 1 Year
Manufacturing Place: INDIA
Battery: AAA 1.5V x2
Packing: Packed in foam

Figure 3: Sample product label with a QR code generated by the AI-based object detection system. The QR code encodes metadata such as product model, serial number, manufacturing date, batch ID, and RoHS compliance, enabling automated traceability and verification.

6. **Batch Tracking and Management :** To organize production into meaningful units, we implemented batch tracking logic:

Every 100 products of the same type are assigned to a new batch (e.g., BATCH01, BATCH02).

The batch number is computed using the product count.

This batch ID appears on both the label and the log, helping in:

Identifying product groups during recalls or testing.

Separating production shifts and tracking cycle efficiency.

This logic reflects real-world batch management and is crucial for large-scale traceability systems.

7. **Operator Interface and User Control:** The system is built to be operator-friendly:

After each labeling cycle, the operator is prompted whether to continue.

If "no" is entered, the loop exits gracefully, saving all data and releasing the webcam.

If "yes," the next detection round begins immediately.

Visual feedback is given for:

Product detected

RoHS compliance status

Label generation

QR display and file path

This ensures that the operator is never left guessing - they always see what the system is doing, building confidence and control into the workflow.

5 Conclusions

This project successfully demonstrates the development of an intelligent, automated system for product traceability and RoHS compliance verification using AI and computer vision. By integrating a YOLOv8-based object detection model with a real-time labeling workflow, we created a seamless pipeline that detects electronic products, verifies their environmental compliance status using structured metadata, and generates QR-based labels containing all relevant manufacturing details. The system goes beyond basic detection by adding layers of regulatory verification, digital labeling, and trace logging - all of which are crucial for modern manufacturing environments. The use of Excel-based RoHS metadata allows for flexible updates, while the visual and QR code labels ensure that every product is uniquely identified, auditable, and compliant. In general, the implementation offers a scalable and adaptable solution that can be extended to more products, connected to physical sensors, and integrated into industrial production lines. Reduce human error, improve transparency, and enable manufacturers to meet regulatory demands more efficiently. By merging AI with compliance-focused automation, this project lays a solid foundation for the future of smart manufacturing and sustainable electronics production.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Dr. Pradeep C for his invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning and natural language processing and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work. []

References

[1] BOCHKOVSKIY, A., WANG, C. Y., AND LIAO, H. Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).

[2] BRADSKI, G. The opencv library. *Dr. Dobb's Journal of Software Tools* (2000). Retrieved from https://opencv.org.

[3] CLARK, A. *Pillow: Python Imaging Library (PIL Fork) Documentation*, 2023. Retrieved from https://pillow.readthedocs.io.

[4] GOOGLE. *Google Colaboratory*, 2023. Retrieved from https://colab.research.google.com.

[5] HARRIS, C. R., MILLMAN, K. J., VAN DER WALT, S. J., ET AL. Array programming with numpy. *Nature 585* (2020), 357–362. 10.1038/s41586-020-2649-2.

[6] ISO/IEC. Iso/iec 18004:2015 - qr code bar code symbology specification, 2015. Retrieved from https://www.iso.org/standard/62021.html.

[7] LIN, K. *qrcode: QR Code image generator for Python*, 2023. Retrieved from https://pypi.org/project/qrcode.

[8] PANDAS DEVELOPMENT TEAM, T. *pandas: Data structures for statistical computing in Python*, 2023. Retrieved from https://pandas.pydata.org.

[9] UNION, E. Directive 2011/65/eu on the restriction of the use of certain hazardous substances (rohs) in electrical and electronic equipment, 2011. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32011L0065.

A Main code sections for the solution

A.1 Install dependencies

Add YOLO, QR code, Excel, and image libraries :

```
!pip install -q ultralytics qrcode pillow openpyxl
```

A.2 Import libraries

Load all required Python and Colab utilities :

```
from ultralytics import YOLO
from IPython.display import display, Javascript, Image as IPyImage
from google.colab.output import eval_js
from PIL import Image, ImageDraw, ImageFont
import qrcode
import io, base64, random, datetime, os, csv
import pandas as pd
```

A.3 Load YOLO model

Use the trained YOLO model for product detection from webcam images :

```
from google.colab import files
print("Please upload the Excel file for RoHS compliance:")
uploaded = files.upload()
```

A.4 Upload RoHS Excel

User uploads an Excel file with RoHS compliance data :

```
excel_filename = next(iter(uploaded))
xls = pd.ExcelFile(excel_filename)
rohs_df = pd.read_excel(xls, sheet_name=0)
rohs_df.columns = rohs_df.columns.str.strip()
```

A.5 Read RoHS data

Load the Excel sheet and clean headers for accurate lookup :

```
rohs_map = dict(zip(rohs_df['Product'].str.upper(), rohs_df['RoHS_Compliant'].str.
                                          lower()))
```

A.6 Build RoHS dictionary

Convert the Excel sheet into a lookup dictionary to match product names with RoHS status:

```
model = YOLO("/content/drive/MyDrive/YOLO_Models/best.pt"
```

A.7 Product metadata

Define static details for each product type: model, specs, components, and suppliers :

```
product_details = {
    'REMOTE': {
        'Model': 'REM0021',
        'Warranty': '1 Year',
        'Manufacturing Place': 'INDIA',
        'Battery': 'AAA 1.5V x2',
        'Packing': 'Packed in foam',
        'Components': [
            "Microcontroller", "IR LED", "Plastic casing", "PCB board",
            "buttons", "battery", "crystal oscillator", "Resistor/capacitor"
        ],
        'Suppliers': [
            "Microchip", "Kingbright PVD", "PCBWay", "Omron",
            "Panasonic", "Panasonic", "Abracon", "Yageo"
        ]
    },
    'LAMP': {
        'Model': 'LA20F',
        'Warranty': '1 Year',
        'Manufacturing Place': 'INDIA',
        'Voltage': '5V DC',
        'Wattage': '3W',
        'Packing': 'Wrapped with bubble sheet',
        'Components': [
            "LED Bulb", "LED Driver Circuit", "Arm Hinge", "USB Cable / Adapter",
            "Diffuser", "Heat Sink", "Base plate"
        ],
        'Suppliers': [
            "Osram", "Inventronics", "Alps", "HAM",
            "Custom Supplier", "INTEX", "BrightTech Supplies"
        ]
    },
    'HARD DISK': {
        'Model': 'SRD00F1',
        'Warranty': '2 Years',
        'Manufacturing Place': 'INDIA',
        'Storage': '500GB',
        'Packing': 'Boxed with silica gel',
        'Components': [
            "Platters", "Spindle Motor", "Read/Write Head", "Head Actuator Arm",
            "Voice Coil Motor", "Controller PCB", "Interface Controller", "Cache
                                                    Memory",
            "Shock Sensor", "Power Connector", "Data Connector", "Enclosure (
                                                    Casing)",
            "Anti-vibration Pads", "EMI Shielding"
        ],
        'Suppliers': [
            "Seagate", "Nidec Corporation", "TDK", "ALPS Electric", "ALPS Electric
                                                    ", "Seagate",
            "Marvell", "Samsung", "STMicroelectronics", "Molex", "Amphenol",
            "Custom Metal OEM Manufacturers", "3M", "Laird"
        ]
    }
}
```

A.8 Batch counters

Track count of each product and assign batch IDs after every 100 pieces :

```
product_counts = {"REMOTE": 0, "LAMP": 0, "HARD DISK": 0}
```

A.9 Auto shift operator ID

Determine the current shift (A, B, or C) and operator ID based on the system time :

```
def get_shift_and_operator():
    now = datetime.datetime.now().hour
    if 6 <= now < 14:
        return "Shift A", "OP_A001"
    elif 14 <= now < 22:
        return "Shift B", "OP_B001"
    else:
        return "Shift C", "OP_C001"
```

A.10 Generate label info

Create a unique serial number, batch ID, device ID, and check RoHS compliance from the uploaded Excel. Combine with product metadata :

```
def generate_label_info(product_name):
    product_counts[product_name] += 1
    batch_num = (product_counts[product_name] - 1) // 100 + 1
    batch_id = f"BATCH{batch_num:02d}"
    date_str = datetime.datetime.now().strftime("%Y%m%d")
    unique_id = str(random.randint(1000, 9999))
    prefix = {
        "HARD DISK": "HDD",
        "LAMP": "LMP",
        "REMOTE": "RMT"
    }.get(product_name, "PRD")
    serial = f"{prefix}-{date_str}-{unique_id}"

    label_date = datetime.datetime.now().strftime("%d-%m-%Y")
    device_id = "DEV" + str(random.randint(1000, 9999))

    rohs_status = rohs_map.get(product_name.upper(), "no")
    rohs = "Compliant" if rohs_status == "yes" else "Not Compliant"

    base = {
        "Product": product_name,
        "Serial Number": serial,
        "Manufacturing Date": label_date,
        "Device ID": device_id,
        "Batch ID": batch_id,
        "RoHS": rohs
    }

    return {**base, **product_details[product_name]}
```

A.11 Capture image

Use webcam to capture an image for AI-based product detection :

```
def take_photo(filename='realtime.jpg', quality=0.8):
    js = Javascript('''
        async function takePhoto(quality) {
            const div = document.createElement('div');
            const capture = document.createElement('button');
            capture.textContent = 'Capture';
            div.appendChild(capture);

            const video = document.createElement('video');
            video.style.display = 'block';
            const stream = await navigator.mediaDevices.getUserMedia({video: true}
                                                     );
            document.body.appendChild(div);
            div.appendChild(video);
            video.srcObject = stream;
            await video.play();

            google.colab.output.setIframeHeight(document.documentElement.
                                                     scrollHeight, true);
            await new Promise((resolve) => capture.onclick = resolve);

            const canvas = document.createElement('canvas');
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            canvas.getContext('2d').drawImage(video, 0, 0);
            stream.getTracks().forEach(track => track.stop());
            div.remove();
            const dataUrl = canvas.toDataURL('image/jpeg', quality);
            return dataUrl;
        }
    ''')
    display(js)
    data = eval_js("takePhoto({})".format(quality))
    binary = io.BytesIO(base64.b64decode(data.split(',')[1]))
    img = Image.open(binary)
    img.save(filename)
    return filename
```

A.12 Setup CSV

Create a new CSV file (if not already present) to store traceability and labeling records :

```
csv_path = "/content/traceability_log2.csv"
if not os.path.exists(csv_path):
    with open(csv_path, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([
            "Timestamp", "Operator ID", "Shift", "Label Status", "QR Path",
            "Product", "Serial Number", "Manufacturing Date", "Device ID", "Batch
                                          ID", "RoHS",
            "Model", "Warranty/Storage", "Manufacturing Place", "Extra Info (
                                          Packing Details)",
            "Components", "Suppliers"
        ])
```

A.13 Main loop

Detect product - generate label - create QR code - print label - log data to CSV - repeat if needed :

```
while True:
    shift, operator_id = get_shift_and_operator()
    print(f"\nCapturing image from webcam... ({shift}, Operator: {operator_id})")
    image_path = take_photo()
    print(f"Image captured: {image_path}")

    results = model.predict(source=image_path, conf=0.5)

    for r in results:
        if len(r.boxes) == 0:
            print("No product detected.")
            continue

        for box in r.boxes:
            cls_id = int(box.cls[0])
            label = model.names[cls_id]
            print(f"Detected: {label}")

            label_data = generate_label_info(label)

            if "Compliant" in label_data["RoHS"]:
                print(f"{label} is RoHS Compliant.")
            else:
                print(f"{label} is NOT RoHS Compliant.")

            qr_content = '\n'.join([f"{k}: {v}" for k, v in label_data.items()])
            qr = qrcode.make(qr_content).resize((200, 200))
            qr_path = f"/content/qr_{label_data['Serial Number']}.png"
            qr.save(qr_path)

            label_img = Image.new("RGB", (600, 400), "white")
            draw = ImageDraw.Draw(label_img)
            font = ImageFont.load_default()
            y = 10
            for k, v in label_data.items():
                if isinstance(v, list): continue
                draw.text((10, y), f"{k}: {v}", fill="black", font=font)
                y += 20
            label_img.paste(qr, (380, 180))

            label_path = f"/content/label_{label_data['Serial Number']}.png"
            label_img.save(label_path)
            display(IPyImage(filename=label_path))
            print(f"Label saved: {label_path}")

            timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            extra_info = f"Packed ({label_data.get('Packing', '')})"
            status = "Success"

            row = [
                timestamp, operator_id, shift, status, qr_path,
                label_data["Product"], label_data["Serial Number"], label_data["
                                                  Manufacturing Date"],
                label_data["Device ID"], label_data["Batch ID"], label_data["RoHS"
                                                  ],
                label_data.get("Model", ""),
                label_data.get("Warranty", label_data.get("Storage", "")),
                label_data.get("Manufacturing Place", ""),
                extra_info,
                ", ".join(label_data.get("Components", [])),
                ", ".join(label_data.get("Suppliers", []))
            ]
            with open(csv_path, mode='a', newline='') as file:
                csv.writer(file).writerow(row)
            print("Logged to CSV.")

    again = input("Detect another product? (yes/no): ").strip().lower()
    if again != "yes":
        break
```

A.14 Download CSV

Allow user to download the complete traceability log after processing :

```
files.download(csv_path)
```

B Product Traceability Search by Device ID, Shift, Batch, Date, or Operator

B.1 Load the CSV log

The CSV file is loaded into a DataFrame using pandas. This file contains all the past traceability records including product info, serial numbers, and QR paths :

```
import pandas as pd
from IPython.display import display, Image as IPyImage
csv_path = "/content/traceability_log2.csv"
df = pd.read_csv(csv_path)
```

B.2 Prompt user to choose trace method

The user is shown a list of 5 ways to trace data: by Device ID, Shift, Batch ID, Manufacturing Date, or Operator ID.

The user selects an option by typing the corresponding number :

```
print("Select trace method:")
print("1. Device ID")
print("2. Shift")
print("3. Batch ID")
print("4. Manufacturing Date (e.g., 06-07-2025)")
print("5. Operator ID")
choice = input("Enter 1, 2, 3, 4, or 5: ").strip()
```

B.3 Map to column name

The input number is mapped to the actual column name used in the CSV file using a dictionary.

This allows easy filtering later :

```
key_map = {
    '1': "Device ID",
    '2': "Shift",
    '3': "Batch ID",
    '4': "Manufacturing Date",
    '5': "Operator ID"
}
key = key_map.get(choice)

if not key:
    print("Invalid selection.")
    exit()
```

B.4 Ask user for the value to trace

Based on the selected method, the user is asked to enter the specific value (e.g., a Device ID or date).

This value will be used to search in the selected column :

```
value = input(f"Enter {key}: ").strip()
```

B.5 Filter rows

The DataFrame is filtered to only include rows where the selected column matches the input value.

This finds all traceability records related to the user's search :

```
results = df[df[key] == value]
```

B.6 Show result

If matching records are found, the table of results is displayed using display() :

```
if not results.empty:
    print(f"\nTraceability records for {key}: {value}")
    display(results)
```

B.7 Show QR for each match

For every matched row, the corresponding QR code image is loaded and displayed using its saved file path in the column "QR Path" :

```
for _, row in results.iterrows():
        print(f"\nProduct: {row['Product']} | Serial: {row['Serial Number']}")
        display(IPyImage(filename=row["QR Path"]))
else:
    print(f"No traceability records found for {key}: {value}")
```

C YOLOv8n Model Training on Custom Dataset

This code imports necessary libraries and trains a YOLOv8 object detection model using a custom dataset defined in data.yaml for 30 epochs with 640x640 image size :

```
from ultralytics import YOLO

model = YOLO("yolov8n.pt")
model.train(data="/content/dataset/data.yaml", epochs=30, imgsz=640)
```