WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AI-Based E-Waste Detection and Intelligent Recycle Recommendation System

Kaviyarasan R, Gowtham S, Kathir U S, Madhusuthanan P, Manikandan G, Ms. Sasikala P

B.E. Students, Department of Computer Science and Engineering
Assistant Professor (Project Supervisor), Department of Computer Science and Engineering
Sri Shakthi Institute of Engineering and Technology (Autonomous), Coimbatore – 641062, Tamil Nadu, India
e-mail: manikandang9355@gmail.com

ABSTRACT

Electronic waste (e-waste) is one of the fastest-growing waste streams, and many discarded devices are thrown away without checking whether they can be repaired, reused, refurbished or dismantled for recoverable components. Traditional identification of e-waste is manual, and the advice available to users is generic. This paper presents an AI-based e-waste detection and intelligent recycle recommendation system. A user uploads an image of an electronic item, and a custom-retrained YOLOv8s model served through Roboflow-hosted inference returns the predicted class and a confidence score. The Google Gemini API then converts the detected item and component catalog information into item-specific lifecycle guidance covering repair, reuse, refurbishment, component recovery and recycling, together with hazard, data-security and safe-disassembly precautions. The system is implemented as a web application with a React, Vite and Tailwind CSS front end, a FastAPI back end and a PostgreSQL database accessed through SQLAlchemy, and it keeps a history of detections. The platform is a decision-support tool: confidence scores are model scores, and image-based recommendations cannot certify the condition or safety of a device. By encouraging extended product life and responsible recycling, the system supports sustainable e-waste management.

Keywords: E-waste detection, YOLOv8s, Roboflow, Generative AI, Gemini API, Lifecycle recommendation, FastAPI, React, PostgreSQL

1. INTRODUCTION

Electronic waste, or e-waste, includes discarded computers, keyboards, mice, mobile phones, chargers, printers and circuit boards. Rapid adoption and short replacement cycles are increasing the amount of e-waste generated [3], and improper disposal wastes valuable materials and can release hazardous substances. Many users do not know what an old device is in recycling terms, whether it can be repaired or reused, or how it should be handled safely.

Object-detection models such as YOLO can identify objects in images quickly [1], [2], which makes them suitable for recognising electronic items from photographs. However, a class label alone does not tell a user what to do next. Generative AI models can bridge this gap by producing understandable, item-specific guidance from the detected label and supporting information [8].

This work proposes a system that combines both: a custom-retrained YOLOv8s model identifies the supported item, and the Gemini API generates lifecycle recommendations, while a PostgreSQL database stores detection records and a component catalog. The main contributions are:

- An end-to-end web application that detects supported e-waste items from uploaded images and reports a confidence score.
- Integration of a generative AI model to turn detections into lifecycle guidance: repair, reuse, refurbishment, recovery and recycling.
- A component catalog and safety module covering recoverable components, hazards, data security and safe disassembly.
- Persistent storage of detection records and a history view built on PostgreSQL and SQLAlchemy.

2. LITERATURE REVIEW

E-waste is a growing global problem, and assessments such as the Global E-waste Monitor report that the amount generated is rising while a large share is not formally collected and recycled [3]. In India, the E-Waste (Management) Rules, 2022 provide a regulatory framework that includes Extended Producer Responsibility [4]. Regulation alone, however, does not help an individual user decide what to do with a particular device.

Object detection identifies and locates objects in images. Two-stage detectors such as Faster R-CNN first propose regions and then classify them [5], whereas YOLO treats detection as a single regression problem and is fast enough for interactive use [1]. YOLOv8 is available in several sizes, and the

small variant (YOLOv8s) balances cost and accuracy [2]. Transfer learning allows a pretrained model to be adapted to a new task with a smaller dataset [6], which is how a general detector can be retrained for e-waste classes. Detection tools commonly stop at a label; the present work pairs a custom-retrained detector with generative lifecycle recommendations and a component catalog in one deployable web application.

3. EXISTING SYSTEM AND LIMITATIONS

Current e-waste identification and disposal rely on the owner's own judgement, manual sorting by collectors and generic recycling advice. Its main limitations are:

- Manual identification that is slow and depends on the skill of the person.
- Users cannot easily tell whether an item can be repaired, reused or refurbished.
- Recycling advice is generic rather than specific to the item.
- Little information on recoverable components and materials.
- Safety and data-security precautions are not given at the right time.
- No record of earlier identifications or recommendations.

4. PROPOSED SYSTEM AND METHODOLOGY

4.1 System Architecture

The system has four layers. The presentation layer is a React application built with Vite and styled with Tailwind CSS, providing image upload, result, dashboard and history views. The application layer is a FastAPI service (Python) that validates uploads, calls the detection and recommendation services and coordinates database operations [9]. The AI layer consists of the custom-retrained YOLOv8s model served through Roboflow-hosted inference (model e-waste-dataset-r0ojc/43) [7] and the Google Gemini API [8]. The data layer is PostgreSQL (database ewaste_db) accessed through SQLAlchemy [10], [11], with the tables detections, component_catalogs and component_items.

4.2 Image Input and Validation

The input is a single image of an electronic item. The backend checks that a file is present, that it is an accepted image type, that its size is within the allowed limit and that it can be processed by the inference service. Invalid input returns an error message.

4.3 Workflow

Figure 1 shows the workflow. The user uploads an image through the React interface and FastAPI validates it; invalid files return an error. The valid image is sent to the Roboflow-hosted YOLOv8s model, which returns the predicted class and confidence score. If no supported object is detected or a service fails, a message is shown instead of an invented result. For a supported item, the backend retrieves component catalog information and requests lifecycle guidance from Gemini, saves the detection record in PostgreSQL and returns the results to the dashboard.

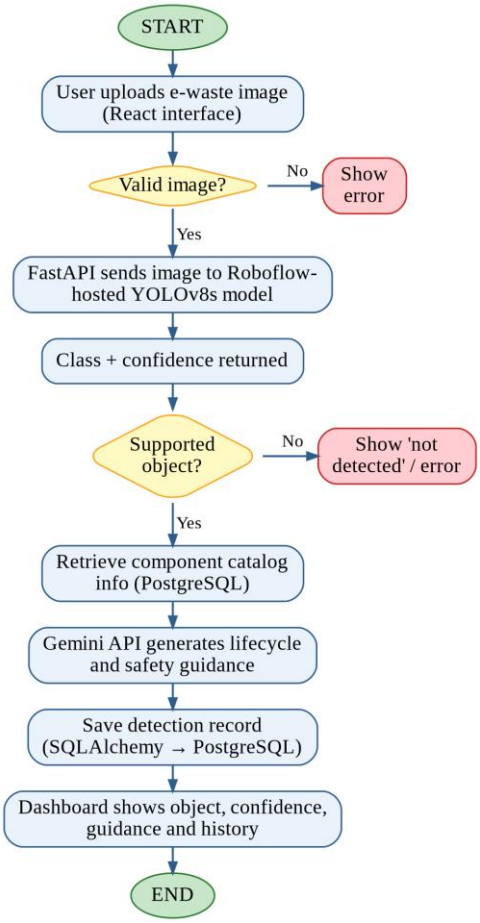


Fig. 1. Workflow of the proposed system

4.4 YOLOv8s-Based E-Waste Detection

YOLO (You Only Look Once) predicts classes and locations in a single pass through the network [1]. The ‘s’ in YOLOv8s denotes the small variant [2]. The model used here was retrained on an e-waste dataset and is hosted by Roboflow, so the backend sends the image to the hosted service instead of loading model weights itself [7]. Training and inference are different processes: uploading an image does not retrain the model. The confidence score is a model score and not a guarantee that the prediction is correct; for example, an earlier test returned Computer-Mouse with a confidence of 58.4%.

4.5 Generative AI Lifecycle Recommendations

The detected class, the confidence score and relevant catalog information are passed to the Gemini API as a structured prompt [8]. The model returns concise guidance such as checking a cable or switch for repair, continuing to use a working device, refurbishing or upgrading, recovering reusable parts, or using an authorised recycler. The system prioritises extending the useful life of a product where safe and practical. Because an image cannot reveal internal faults, these recommendations are decision-support guidance, may be incomplete, and should be constrained by catalog data and safety rules.

4.6 Component Catalog, Safety Guidance and Data Management

The component catalog links a device category to its components, their recovery possibilities and handling information. The safety module explains battery and electrical hazards, safe handling and storage, dismantling precautions and data-security steps before disposal; for example, a swollen lithium-ion battery should not be dismantled by an untrained user. Detection records are stored in PostgreSQL so that previous detections can be reviewed in the history view.

4.7 Algorithm

- Initialise the front end, API, database connection and the Roboflow and Gemini configuration.
- Receive the uploaded image and validate its type and size; on failure, return an error.
- Send the valid image to the YOLOv8s model and obtain the class and confidence score.

- Retrieve component catalog and recovery information for the detected class.
- Generate lifecycle and safety guidance using the Gemini API.
- Store the detection record in PostgreSQL.
- Display the object, confidence, recommendations, safety guidance and history.

5. TECHNOLOGY STACK AND REQUIREMENTS

Layer	Technologies
Front end	React, Vite, Tailwind CSS, JavaScript, HTML5/CSS3
Back end	Python, FastAPI, Uvicorn, REST APIs
AI services	YOLOv8s (custom-retrained), Roboflow hosted inference, Google Gemini API
Database	PostgreSQL, SQLAlchemy
Tools	VS Code, Git/GitHub

Table 1. Technology stack

The system runs on Windows, Linux or macOS on a 64-bit machine with an Intel Core i3 or higher processor, at least 4 GB RAM (8 GB recommended), 256 GB storage and a stable internet connection, which is required for hosted inference and the Gemini API. A modern browser such as Chrome or Edge is used.

6. RESULTS AND DISCUSSION

The system was implemented as a working web application, and during final validation the main modules were reviewed: image processing, display of detected items with confidence scores, lifecycle and recycling recommendations, database records, component recovery information, safety guidance and detection history. The user first uploads an image (Fig. 2) and then views the detection result (Fig. 3).

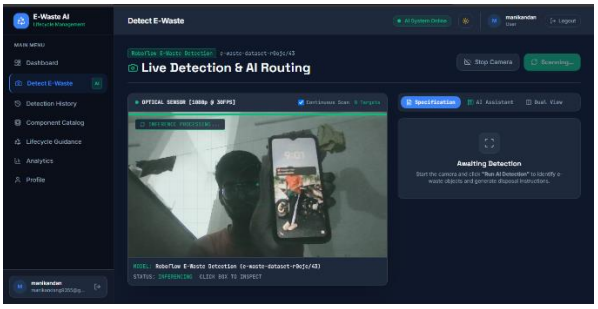


Fig. 2. Image upload interface

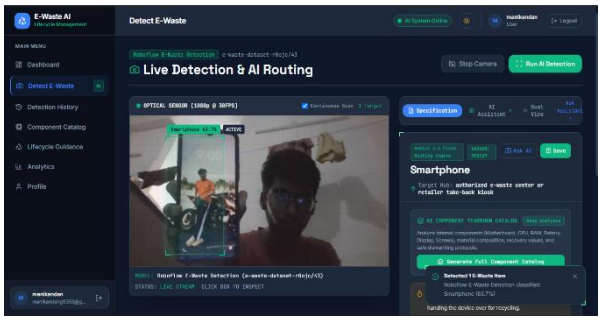


Fig. 3. Detection result with confidence score

The recommendation view (Fig. 4) presents lifecycle guidance for the detected item together with component recovery information, helping the user decide between repair, reuse, refurbishment, recovery and recycling.

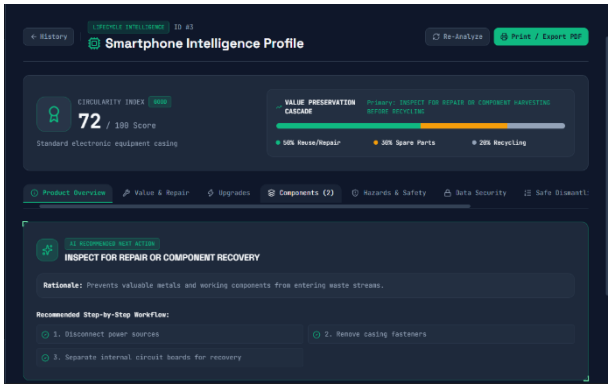


Fig. 4. Lifecycle recommendations and component recovery

The safety module (Fig. 5) presents hazard, data-security and safe-disassembly guidance for the detected device. The detector should be evaluated on a test dataset using precision, recall, mAP and inference time; Table 2 lists these metrics.

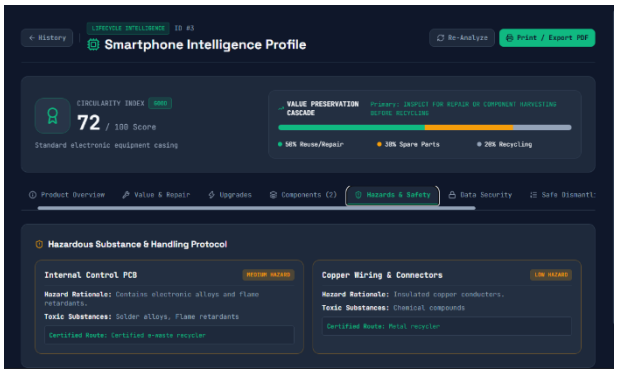


Fig. 5. Safety and hazard guidance

Metric	Description	Value
Precision	Share of predicted detections that were correct	0.92
Recall	Share of actual objects that were detected	0.89
mAP	Aggregate detection performance	0.91
Inference time	Time to produce a prediction	2s

Table 2. Detection model evaluation metrics

The history view (Fig. 6) retrieves previously recorded detections from PostgreSQL so that users can review earlier results.

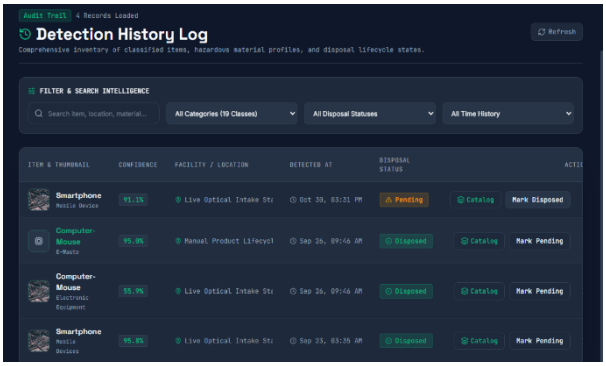


Fig. 6. Detection history

The system separates responsibilities: YOLOv8s identifies the object, Gemini generates recommendations, PostgreSQL stores records and React displays results. Its reliability depends on the classes the model was trained on, image quality, catalog coverage and the accuracy of generated text. A high

confidence score does not guarantee correctness, an image cannot reveal internal faults or electrical safety, and the external services (hosted inference and the Gemini API) may fail, in which case an error message is shown. The platform is therefore a decision-support tool.

7. CONCLUSION AND FUTURE SCOPE

This paper presented an AI-based system that detects supported e-waste items from images using a custom-retrained YOLOv8s model, generates lifecycle recommendations with the Gemini API and stores detection records and a component catalog in PostgreSQL. Built on React, FastAPI and SQLAlchemy, it takes users from a photograph to repair, reuse, recovery or recycling guidance with safety precautions, and encourages extending product life instead of unnecessary disposal.

Future work includes training the detector on more e-waste classes and evaluating it under varied image conditions, letting users supply fault and condition details, integrating a verified directory of authorised recycling centres, estimating environmental impact with documented assumptions, multilingual (including Tamil) recommendations, detecting individual components in dismantled devices, validating recommendations against structured rules and verified catalog data, an analytics dashboard, and local inference to reduce dependence on external services.

ACKNOWLEDGEMENT

The authors thank the management, Principal and the Head of the Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, and the project guide, Ms. Sasikala P, for providing the facilities and support for this work.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[2] G. Jocher, A. Chaurasia and J. Qiu, “Ultralytics YOLOv8,” 2023. [Online]. Available: https://docs.ultralytics.com

[3] C. P. Baldé et al., The Global E-waste Monitor 2024. Geneva/Bonn: International Telecommunication Union and UNITAR, 2024.

[4] Ministry of Environment, Forest and Climate Change, Government of India, “E-Waste (Management) Rules, 2022,” 2022.

[5] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017.

[6] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, 2010.

[7] Roboflow, Inc., “Roboflow documentation: Hosted inference and model deployment.” [Online]. Available: https://docs.roboflow.com

[8] Google, “Gemini API documentation.” [Online]. Available: https://ai.google.dev/gemini-api/docs

[9] S. Ramírez, “FastAPI documentation.” [Online]. Available: https://fastapi.tiangolo.com

[10] The PostgreSQL Global Development Group, “PostgreSQL documentation.” [Online]. Available: https://www.postgresql.org/docs

[11] SQLAlchemy Authors, “SQLAlchemy documentation.” [Online]. Available: https://docs.sqlalchemy.org

[12] Meta Open Source, “React documentation.” [Online]. Available: https://react.dev