AI Hardware Project Proposal

Edge-AI Waste Classification on OpenMV H7 (TinyML)

Team Name: Circuit board layout

Students: Mengzi Cheng, Shuai Tu, Sirui You, Xueyi Zhang

Date: November 5, 2025

Platform Selection

We selected the OpenMV H7 as our platform for some reason.

1. OpenMV H7 supports TinyML / TensorFlow Lite Micro, enabling machine-learning inference directly on a microcontroller.
2. The board includes an onboard camera, making it suitable for real-time computer-vision tasks such as waste classification.
3. The platform allows model quantization and real-time performance evaluation, demonstrating Edge-AI deployment and hardware acceleration.

Problem Definition

Most waste-classification systems rely on cloud-based processing, which causes high latency, higher power consumption, and privacy concerns. This project aims to achieve real-time, on-device waste classification on the OpenMV H7 microcontroller using TinyML. The goal is to perform efficient edge inference with low latency, low power usage, and no cloud dependency, while demonstrating model quantization and embedded AI performance.

Technical Objectives

1. Implement on-device image classification on the OpenMV platform
2. Train and deploy a lightweight TinyML model suitable for MCU constraints
3. Achieve basic real-time inference performance (target around 5–10 FPS)
4. Apply model optimization techniques such as quantization or compression

5. Evaluate accuracy, latency, and memory usage on the device

Methodology

1. Hardware Setup

Use the OpenMV H7 microcontroller with an integrated camera for on-device vision processing and inference. Set up basic lighting and object placement environment for waste item testing.

2. Software Tools

Train the model using Edge Impulse or PyTorch and export to TensorFlow Lite Micro format. Use the OpenMV IDE for deployment, firmware development, and debugging.

3. Model Design

Develop a lightweight TinyML classification model suitable for MCU resource limits. Include image preprocessing on device and apply post-training quantization to reduce model size and improve efficiency.

4. Performance Metrics

Measure on-device accuracy, inference latency, frames-per-second (FPS), and memory usage (Flash/RAM). Monitor system responsiveness during real-time operation.

5. Validation Strategy

Test the system under varied lighting conditions, distances, and object orientations. Validate classification performance on real waste items and record results across multiple runs to ensure stability and basic robustness.

Expected Deliverables

1. A functioning edge-AI waste classification system running on the OpenMV platform, demonstrating real-time on-device inference.
2. Source code and model files hosted in a GitHub repository (including training scripts, deployment files, and OpenMV firmware code).
3. Trained TinyML models in both FP32 and INT8 formats, along with model conversion and optimization records.
4. Performance evaluation results including accuracy, FPS, latency, and memory usage.

5. Final technical documentation describing system design, methodology, and evaluation results.
6. Presentation slides for in-class project demonstration.

Team Responsibilities

Name	Role	Responsibilities
Mengzi Cheng	Team Lead	Project planning, coordination and presentation
Shuai Tu	Hardware	OpenMV and system setup, wiring, device configuration, deployment support
Sirui You	Software	Dataset preparation, model training, TinyML conversion, code implementation
Xueyi Zhang	Evaluation & Testing	Performance testing, and final integration, documentation submission

Timeline and Milestones

Week	Milestone	Deliverable
2	Proposal	PDF + GitHub submission
4	Midterm presentation	model quantization and benchmarking, presentation slides
6	Integration & testing	on-device inference, FPS/latency results, robustness testing and analysis
Dec.18	Final presentation	Final report, demo video, slides, GitHub archive

Resources Required

1. OpenMV H7 board
2. Edge Impulse or PyTorch setup
3. Waste dataset (public + self-collected)
4. Laptop for model training and testing

References

- Warden, P., & Situnayake, D. (2020). TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O’Reilly Media.
- Albawi, S. et al. (2017). Understanding CNNs for Edge Vision Tasks. IEEE ICMLA.
- Howard, A. et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861
- Sandler, M. et al. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR.