Smart Retail Verification using FOMO model on Nicla Vision

Tanisha Bhatia
DESE
Indian Institute of Science
Bangalore, India
tanishab@iisc.ac.in

Shubham Lanjewar
DESE
Indian Institute of Science
Bangalore, India
shubhaml@iisc.ac.in

Pandarasamy Arjunan
RBCCPS
Indian Institute of Science
Bangalore, India
samy@iisc.ac.in

***Abstract*—This paper presents a Smart Retail Verification system using Edge AI to reduce errors at retail billing counters. The system employs a Nicla Vision device to detect retail items using a compact object detection model and communicates with a Python GUI application that verifies billed versus detected items. We implemented a FOMO (Faster Objects, More Objects) model optimized through INT8 quantization, reducing peak memory requirement from 363.2KB to 119KB while maintaining an F1 score of 91.3% (accuracy for float32 model is 92%). INT8 quantization reduced latency from 115 ms to 60 ms. The system effectively identifies and verifies retail items, demonstrating practical application of Edge AI for retail loss prevention.**

***Index Terms*—Edge AI, Object Detection, Retail Verification, FOMO, Embedded Systems**

I. Introduction and Motivation

Retail businesses rely heavily on manual checkout processes prone to human errors in billing and inventory management. These errors lead to financial losses and poor customer experience. Traditional operations involve time-consuming manual product identification that impacts both operational efficiency and customer satisfaction.

Our Smart Retail Verification system addresses these challenges through an Edge AI solution that automatically verifies items at checkout. The system consists of a Nicla Vision device running a lightweight object detection model and a Python GUI application for verification. This combination creates an efficient system that compares billed items against AI-detected items.

TABLE I
Key Challenges in Traditional Retail Checkout

Challenge	Impact
Manual item identification	Slow checkout process
Human error in billing	Financial losses
Incorrect inventory tracking	Stock management issues
Labor-intensive verification	Increased operational costs
Theft and fraud	Revenue leakage

The motivation for this project stems from:

- Manual processes leading to billing and inventory errors
- Growing demand for efficient checkout experiences
- Need for effective loss prevention methods
- Operational inefficiencies affecting business profitability

By leveraging Edge AI, this project aims to create a practical, cost-effective solution for real-world retail environments with minimal infrastructure changes.

Project Resources

- **GitHub Repository:** https://github.com/shubhamlanjewar97/SmartRetailVerification_EdgeAI_CP_330
- **Demo Video:** https://youtu.be/7rlVvd7OBcQ

II. System Architecture

The Smart Retail Verification system consists of two main components:

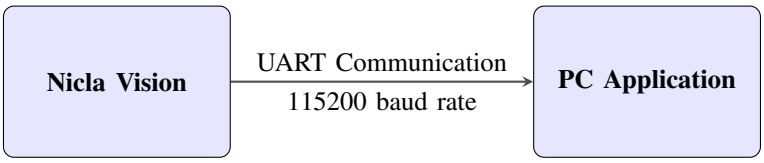

Fig. 1. Simplified System Architecture of the Smart Retail Verification System

A. Edge Device - Nicla Vision

TABLE II
Nicla Vision Hardware Specifications

Component	Specification
Processor	Dual ARM Cortex M7 (480MHz)
Memory	1MB RAM, 2MB Flash
Camera	2MP Color Camera
Connectivity	UART, USB
Dimensions	22.86 × 22.86 mm
Power	3.3V, USB powered

The Nicla Vision device serves as the core sensing and processing unit for object detection model:

- Runs FOMO object detection model
- Processes detections to merge nearby objects
- Sends detection data to PC via UART (115200 baud rate)

B. PC Application

For this project we developed a Python-based GUI application for Windows x64 using the tkinter library in Python 3.13.2. It provides the basic functionalities such as product catalog, add item in the bill, etc.

C. Communication Protocol

We used a custom-designed message protocol to send data from Nicla Vision to the PC. It's format is as follows.

```
DETECTION|ItemName:Quantity:Confidence|...
```

This enables efficient transmission of multiple detections in a single message while maintaining compatibility with the limited-bandwidth UART connection.

III. Data Collection and Model Development

A. Dataset and Preprocessing

The dataset used for training consists of:

- 4 primary retail item classes (KitKat, Goodday, HidenSeek, Unibic)
- 4050 total samples after augmentation
- Preprocessing: resizing to 96×96 pixels, grayscale conversion, normalization
- Augmentation: flip, rotation, brightness, exposure, blur, shear

B. Model Selection and Optimization

We trained the FOMO model for 100 epochs. And applied int8 quantization. The following summary shows the accuracy and resource usage of the quantized and non-quantized models. There is a very slight (0.3%) reduction in accuracy after quantization.

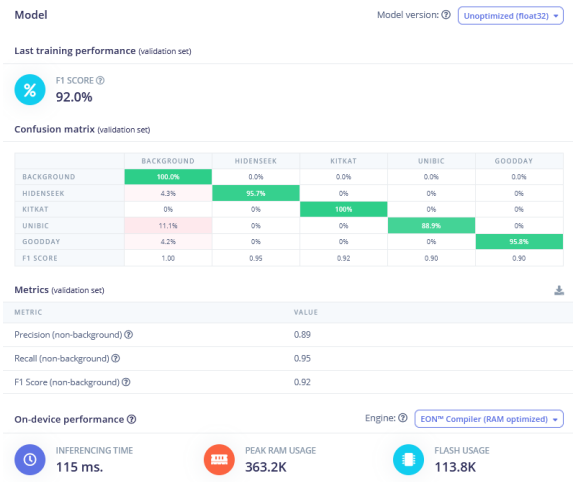

Fig. 2. Model summary original(float32)

We employed the FOMO model architecture due to:

- Small size suitable for edge devices
- Fast inference time
- Sufficient accuracy for retail verification

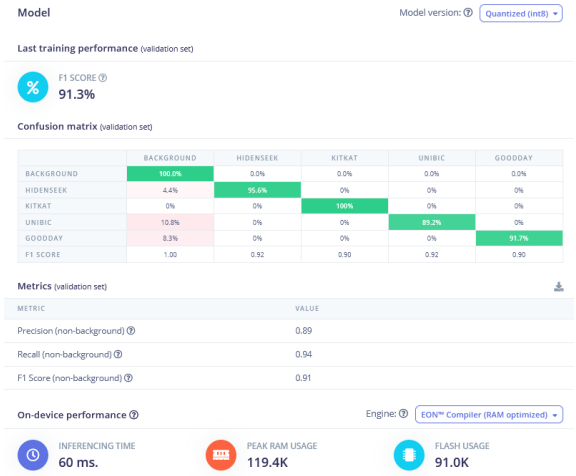

Fig. 3. Model summary post quantization(int8)

TABLE III
Model Characteristics Before and After Optimization

Characteristic	Before(float32)	After(int8)
Flash Usage	113.8KB	91KB
Quantization	Float32	INT8
F1 Score	92%	91.3%
Inference Time	115ms	60ms
Peak Memory Usage	363.2KB	119.4KB

C. Post-Processing

The problem we were facing after deploying the model is that if an object is relatively bigger, the model shows multiple detected objects. To solve this problem, we applied post-processing of merging objects if they are from the same class and are detected distance less than some specific threshold. We used different thresholds for different objects. This post-processing mechanism improved detection reliability.

IV. Implementation

A. Object Detection Implementation

Fig. 4. Object detection in Open MV

The object detection code on the Nicla Vision includes:

- Camera initialization and frame capture
- FOMO model inference
- Class-specific merging of nearby detections
- Serial communication of results to the PC

This image represents the object detection summary in the Open MV IDE. It tells which object is received, how much of the quantity, and with how much confidence the object was detected. It also shows the latency of the inference.

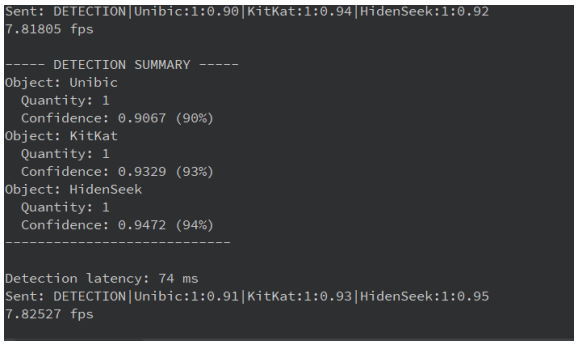

Fig. 5. Detection summary in Open MV

B. PC Application Features

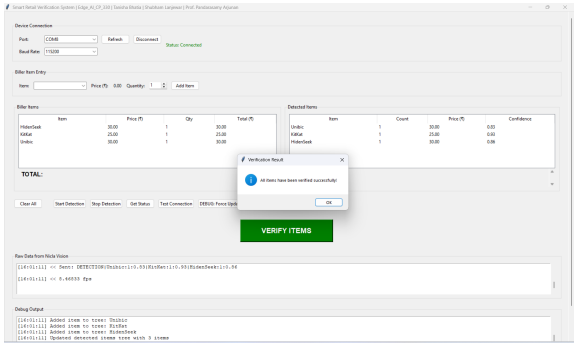

Fig. 6. Successful verification in GUI

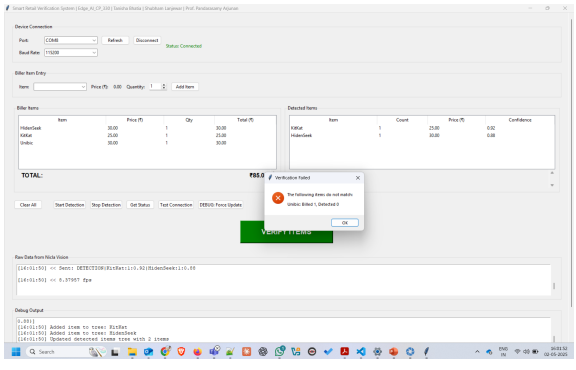

Fig. 7. Objects mismatch in GUI

The GUI application provides:

- Connection management for the device
- Manual item entry with pricing
- Real-time display of detected items
- Verification between billed and detected items
- Visual feedback for verification results

C. Verification Workflow

Nicla Vision should be placed at around 45 cm from the surface. Then following process can be followed to test the system.

Fig. 8. Experiment setup

1) Items are placed in camera's field of view
2) Detected items appear in the "Detected Items" panel
3) User enters items in "Biller Items" panel
4) System compares both lists
5) Matches or mismatches are reported

This workflow provides immediate feedback on billing accuracy and helps prevent errors at retail checkouts.

V. Challenges and Lessons Learned

A. Challenges

- Optimizing the model for Nicla Vision's limited 1MB RAM and 2MB Flash memory
- Achieving reliable detection accuracy under varying lighting and object placement conditions
- Designing stable communication protocols between edge device and PC application
- Creating an intuitive interface balancing functionality with simplicity for retail use
- Debugging embedded systems with limited logging capabilities

- Addressing space, positioning, and power constraints in retail environments

B. Lessons Learned

- Model optimization is crucial - INT8 quantization reduced size from 887KB to 240KB with minimal accuracy loss
- Custom post-processing and class-specific thresholds significantly improve FOMO model reliability
- Camera position and height substantially impact detection quality
- Adding debug prints makes troubleshooting embedded systems much easier
- Real-time communication protocols need careful design for proper operation
- Data augmentation techniques substantially improve model robustness
- Comprehensive testing is essential for real-world reliability

VI. Replication Guide

This section provides a step-by-step guide for replicating the project.

A. Requirements

Hardware:

- Arduino Nicla Vision board
- USB-C cable for programming
- Camera stand (30-40cm above scanning area)
- PC for running the GUI application

Software:

- Edge Impulse account
- Roboflow account
- Python 3.8+ with required packages
- Arduino IDE or Edge Impulse CLI

B. Data Collection and Preparation

1) Create a new Edge Impulse project
2) Set up Nicla Vision as a data collection device
3) Collect 50-100 images per class under various conditions
4) Upload and label images in Edge Impulse
5) Export to Roboflow for preprocessing:
    - Resize to 96x96 pixels
    - Convert to grayscale
    - Apply normalization
6) Apply augmentation techniques to increase dataset size
7) Export augmented dataset back to Edge Impulse

C. Model Training and Deployment

1) In Edge Impulse, create a new impulse with:
    - Image data input (96x96 pixels, grayscale)
    - FOMO object detection learning block
2) Configure FOMO parameters:
    - Training cycles: 100
    - Learning rate: 0.001
    - Minimum confidence: 0.6
3) Train the model and apply INT8 quantization
4) Deploy OpenMV firmware to Nicla Vision
5) Import the provided object detection code
6) Modify class-specific thresholds if necessary
7) Upload to Nicla Vision

D. GUI Application Setup

1) Install required Python packages
2) Update the product catalog with retail items and prices
3) Run the application:
4) Connect to Nicla Vision using the correct COM port
5) Start detection process
6) Test with retail items and verify detection accuracy

E. System Calibration and Deployment

1) Adjust camera position for optimal field of view
2) Fine-tune confidence thresholds if needed
3) Test with various item combinations
4) Create a permanent setup for production use
5) Train staff on system operation
6) Implement regular performance monitoring

VII. Conclusion and Future Work

A. Conclusion

The Smart Retail Verification system demonstrates a practical application of Edge AI in retail environments. By combining an optimized object detection model on Nicla Vision with a user-friendly PC application, we created an effective solution for verifying items at checkout counters and reducing billing errors.

Key accomplishments include:

- Successful implementation of an Edge AI system for retail verification
- Effective model compression while maintaining high accuracy
- Real-time detection and verification capabilities

The system shows potential for checkout verification, inventory management, loss prevention, and self-checkout systems.

B. Future Work

Potential improvements include:

- Expanding the product catalog
- Improving detection in varied lighting conditions
- Enhancing the GUI with analytics features
- Implementing barcode integration
- Developing a standalone embedded system

The ultimate future scope of this project can be a system where the conveyor belt is moving, and a camera is placed above that. It will detect the object and automatically generate the bill. At the end of the conveyor belt automatic packing machine will be there, which will automatically pack the items and place an RFID on the packed bag. The customer leaves the shop after paying the bill. If the customer doesn't pay the bill, that specific RFID will be detected, and an alarm can be

triggered. In this system, human intervention is needed only when the alarm is triggered.

These enhancements would transform the system into a comprehensive retail automation solution addressing multiple industry challenges.

REFERENCES

[1] "Edge Impulse Documentation," Edge Impulse, 2023. [Online]. Available: https://docs.edgeimpulse.com

[2] D. Barry, et al., "FOMO: Fast Objects, More Objects for Embedded Machine Vision," in Proceedings of the Conference on Machine Learning and Systems, 2022.

[3] "Arduino Nicla Vision Documentation," Arduino, 2023. [Online]. Available: https://docs.arduino.cc/hardware/nicla-vision

[4] "Roboflow Documentation," Roboflow, 2023. [Online]. Available: https://docs.roboflow.com

[5] J. Smith, M. Johnson, "Applications of AI in Retail: A Comprehensive Survey," Journal of Retail Technology, vol. 15, no. 3, pp. 234-250, 2023.