Smart-Retail-Product-Detection-and-Analysis-Using-YOLOv8

By Khushi Rawat 23BCE7131, Dhani 24BCE7569, Ramsha Annam 23BCE8215

Abstract

The challenges faced by departmental stores like D-Mart, Big Bazaar, Jio Mart, etc., to track human activities within physical spaces are significant. Right now, stores have very limited access to tracking customer interactions, relying mostly on billing transactions, leaving them unaware of what the customer looked at, touched, or considered buying. The main objective of this project is to fill the gap between store management and customer behavior. It includes detailed data on customer movement within the store. It tracks the interaction between products and customers, such as which products they picked from where and where they kept them. It also tracks how much time they spend in each section and keeps data on customer flow throughout the store while analyzing traffic patterns. When customer behavioral data is integrated with existing sales records, it helps store management modify the store layout to increase customer engagement effectively. It is also highly effective for store security as it identifies illicit activities and provides alert warnings to the management, allowing immediate actions to counter major or minor incidents. By enhancing the understanding of customer engagement, it helps improve the operational efficiency of the store and elevate the customer experience. This includes knowledge of product placement and store layout based on live feed (real-time customer behavior). Through report analysis, staff can be strategically positioned to assist customers in all situations, such as high-traffic areas, ensuring their security, helping them find what they need, and assisting physically challenged customers. This will lead to increased customer satisfaction, which will ultimately prove profitable for the store. Stores will earn a good image in the market, increase sales, and continue improving customer facilities. The main goal is to empower the retail data management system of stores to make data-driven decisions that will be profitable for the store and convenient for the customers.

Introduction: Bridging the Gap Between Management and Customer Experience

Retail businesses face a significant challenge in understanding and improving the customer experience within physical store spaces. While departmental stores like D-Mart, Big Bazaar, and Jio Mart can track sales transactions, they lack insights into customer interactions before a purchase is made. What products did customers pick up, examine, and return? What caught their attention but didn’t make it to the billing counter? These unanswered questions highlight a gap between store management and customer behavior—a gap this innovative project aims to bridge. The goal of this project is to empower retail data management systems by capturing, analyzing, and leveraging customer behavioral data within stores. By monitoring customer movements and interactions in real-time, this system provides detailed insights into their engagement with different products and sections. For instance, it tracks which items customers pick up, where they place them back, and the time spent in specific areas. These behavioral data points, when combined with existing sales records, can enable stores to make data-driven decisions about product placement, layout optimization, and personalized service strategies. The result? Increased customer engagement, satisfaction, and profitability. Beyond customer experience enhancement, this project also strengthens security measures within stores. By identifying suspicious or illicit activities through behavioral monitoring, the system alerts management in real time, allowing immediate action to prevent incidents. Moreover, strategic staff positioning based on traffic patterns ensures high-touch customer service in crowded areas, improving assistance for shoppers, especially those who may face challenges, such as physically disabled customers. This advanced system not only elevates operational efficiency but also reshapes the perception of retail spaces. Stores can enhance their brand image in the competitive market, boost sales, and continuously refine customer facilities. By integrating cutting-edge technologies such as IoT sensors, computer vision, and AI algorithms, this project positions stores as innovative leaders that prioritize customer satisfaction and safety. Ultimately, this initiative transforms retail spaces into data-driven environments where every decision—from store layout to staff allocation—is informed by customer behavior insights. It bridges the gap between management and customer experiences, creating a win-win scenario for both stakeholders. With improved facilities and personalized service, customers find shopping convenient and enjoyable, while stores gain valuable insights to thrive in an ever-evolving market.

Literature Review: AI-Driven Computer Vision for Retail Analytics

Retail businesses are constantly evolving to improve customer experiences and operational efficiency. With advancements in computer vision and artificial intelligence, retailers can now analyze customer behavior through CCTV footage and automated tracking systems. This project focuses on utilizing computer vision techniques for object detection, activity recognition, and customer behavior analysis to enhance store layout and merchandising strategies. The implementation of AI-driven analytics in retail environments not only improves efficiency but also provides a deeper understanding of consumer preferences, leading to increased customer satisfaction and profitability. The core of this project is based on computer vision techniques, specifically object detection and tracking. Several state-of-the-art models have been developed in this field, including YOLO v8 (You Only Look Once), SSD (Single Shot MultiBox Detector), and Faster R-CNN. These models leverage Convolutional Neural Networks (CNNs) to extract features from images and videos, enabling precise localization and classification of objects. Each object detection model has its own advantages and trade-offs in terms of accuracy, speed, and computational efficiency. YOLO v8 is known for its real-time detection capabilities, providing high-speed processing while maintaining good accuracy. SSD strikes a balance between speed and accuracy, making it suitable for embedded systems and mobile devices. Faster R-CNN provides high accuracy with region-based proposals, ideal for complex retail environments where precision is paramount. In the retail context, object detection helps in identifying products, tracking customer behavior, and analyzing engagement patterns. Retailers use these insights to determine which products are most frequently examined or purchased, allowing for better inventory management and promotional strategies. However, challenges such as varying lighting conditions, product diversity, and real-world store environments need to be addressed for effective implementation. Adapting object detection models to dynamic retail settings requires continuous model training with diverse datasets that encompass different store layouts, customer demographics, and shopping behaviors. Beyond object detection, customer activity recognition plays a crucial role in understanding shopping behaviors. The primary objective is to identify actions such as “picking up,” “examining,” or “returning” items. Analyzing sequences of interactions with products provides valuable insights into customer interests and preferences. Several advanced models facilitate human activity recognition. Hidden Markov Models (HMMs) are effective for sequential activity analysis, commonly used in speech and gesture recognition. Recurrent Neural Networks (RNNs) are useful for processing time-series data and identifying temporal patterns in customer behavior. Long Short-Term Memory (LSTM), an extension of RNNs, overcomes short-term memory limitations and helps in analyzing long sequences of customer actions. By leveraging video surveillance and deep learning techniques, these models provide accurate activity recognition systems tailored for retail applications. Understanding customer actions can help retailers design better marketing strategies, such as personalized promotions and targeted advertisements. However, differentiating between intentional and unintentional activities remains a significant challenge. For example, a customer might pick up an item only to return it due to an external distraction. Improving model accuracy requires refining training data and incorporating additional contextual cues, such as facial expressions and hand movements, to better classify activities.

Methodology

Introduction to the Problem and Initial Approach

The objective of this project was to build a **real-time product detection system** that could analyze customer behavior by tracking product interactions in a retail store. The goal was to identify grocery items being picked by customers, detect hand movements indicating a grabbing action, and recognize whether the items were placed in a **store basket** or a **trolley.** This analysis aimed to improve product placement strategies and optimize customer service by providing actionable insights.

Challenges with Initial Dual-Model Setup

1. Two-Model Concept and Its Limitations

The initial approach adopted a **two-model strategy** using YOLOv8, where:

- **Model 1** was responsible for detecting grocery items.
- **Model 2** handled the detection of hands, store baskets, and trolleys.

However, this approach led to several complications:

- **High GPU Memory Usage:** Running two models simultaneously resulted in excessive GPU memory consumption, often leading to **out-of-memory (OOM) errors.** To address this, the number of workers was gradually increased to **12**, which initially showed some improvement. However, the problem persisted as training progressed with larger batches.

- **Poor Accuracy and Low mAP:** Despite extensive hyperparameter tuning, the combined models produced poor results, with a final mAP@50 score of **0.0136,** which was too low for real-time applications.
- **Difficulty in Hand-to-Product Association:** Since both models operated independently, associating hand actions with product interactions became inconsistent, making the system unreliable.

2. Transition to a Single-Model Strategy

To overcome these challenges, the approach was modified to a **single-model strategy** by creating a **custom dataset** that included:

- **Grocery Items:** Different types of packaged goods, fresh produce, and other store items.
- **Hands with Grabbing Actions:** To detect whether a product was being picked.
- **Store Basket and Trolley:** To identify where the picked items were placed.

This single-model approach simplified the architecture, reduced computational overhead, and improved the overall accuracy.

Addressing Class Imbalance Through Class Weights

1. Identifying and Correcting Class Imbalance

An initial analysis of the dataset revealed a significant **class imbalance**. Underrepresented classes such as **scrubber, dal, and trolley** had very few samples compared to frequent classes like **butter and masala.** This imbalance skewed the model's learning, causing it to ignore less frequent classes.

To address this, **class weights** were manually assigned to balance the loss function and ensure that the model paid more attention to the underrepresented classes. Higher weights were assigned to classes with lower frequencies to encourage the model to learn these categories more effectively. The class weights were calculated using the inverse of the class frequency:

$$\text{Class Weight}=\frac{1}{\text{Frequency of Class Occurrence}}$$

- **Higher Weights:** Assigned to scrubber, dal, and trolley to prevent underfitting.
- **Lower Weights:** Given to more frequent classes like butter and masala to prevent overfitting.

2. Impact of Class Weights

Introducing class weights significantly improved the model’s performance for underrepresented classes. The mAP values for these classes increased by more than **20%,** enhancing the overall robustness of the system. This adjustment prevented the model from disproportionately favoring dominant classes, ensuring a more balanced performance.

Final Hyperparameter Tuning and Training Process

1. Switching from SGD to AdamW for Better Optimization

The initial models were trained using the **Stochastic Gradient Descent (SGD)** optimizer, which did not yield satisfactory results. After extensive experimentation, the optimizer was switched to **AdamW**, which is known for its adaptive learning rate and better weight decay handling. This change significantly improved the convergence rate and overall stability during training.

2. Incorporating Cosine Learning Rate Decay

To maintain a controlled learning process, a **cosine learning rate decay** was used, which gradually reduced the learning rate over time. This approach helped prevent overfitting and allowed for smooth learning transitions across epochs.

3. Adjusting Loss Gains to Improve Accuracy

Several adjustments were made to the loss function to fine-tune model performance:

- **Box Loss (Position):** The box loss was lowered to prevent overfitting on positional features, improving generalization.
- **Classification Loss (Class Prediction):** The classification loss was increased to ensure better learning of weaker classes.
- **Distribution Focal Loss (DFL):** Distribution focal loss was fine-tuned to further improve class-level accuracy.

Data Augmentation and Training Enhancements

1. Extensive Data Augmentation for Generalization

To improve the model's generalization and avoid overfitting, various augmentation techniques were applied to the training data:

- **Mosaic Augmentation:** Increased dataset variation by combining multiple images into one.
- **Mixup Augmentation:** Combined two images with different weights to encourage the model to generalize better.
- **Flip and Rotation:** Random horizontal and vertical flips were applied to simulate real-world orientations.
- **Hue, Saturation, and Brightness Adjustments:** Subtle variations in color were introduced to improve robustness to lighting changes.
- **Perspective and Scale Changes:** Small distortions and translations were applied to increase variation in object positions.

2. Increasing Epochs and Batch Size for Stability

The initial training was conducted for **75 epochs,** but due to poor convergence, the epoch count was gradually increased to **100 and then 120.** However, further analysis suggested that 100 epochs provided the best balance between training time and model accuracy.

Final Hyperparameter Settings

1. Optimal Learning Rate and Momentum

- **Learning Rate:** Lowered to **0.0003** to prevent aggressive weight updates.
- **Momentum:** Set to **0.95** to stabilize learning and prevent oscillations.

2. Warmup Strategy for Smoother Training

A warmup phase of **8 epochs** was introduced with a high momentum value to allow for smoother transitions into full training.

3. Patience and Early Stopping to Avoid Overfitting

The model was configured with a **patience value of 20 epochs,** ensuring that training stopped early if no improvement was observed, thereby preventing unnecessary computation.

Training Results and Final Performance

After incorporating all these improvements, the final YOLOv8 model demonstrated remarkable enhancements in accuracy and robustness:

- **mAP@50:** Increased from **0.0136** in the initial dual-model setup to **60.8%.**
- **mAP@50-95:** Improved to **46.3%,** indicating better detection across a range of IoU thresholds.
- **Improved Detection for Underrepresented Classes:** mAP for scrubber, trolley, and dal increased by over **20%** due to the addition of class weights.
- **Inference Time:** Reduced to an average of **30ms per frame,** ensuring real-time applicability.

Real-Time Deployment and System Integration

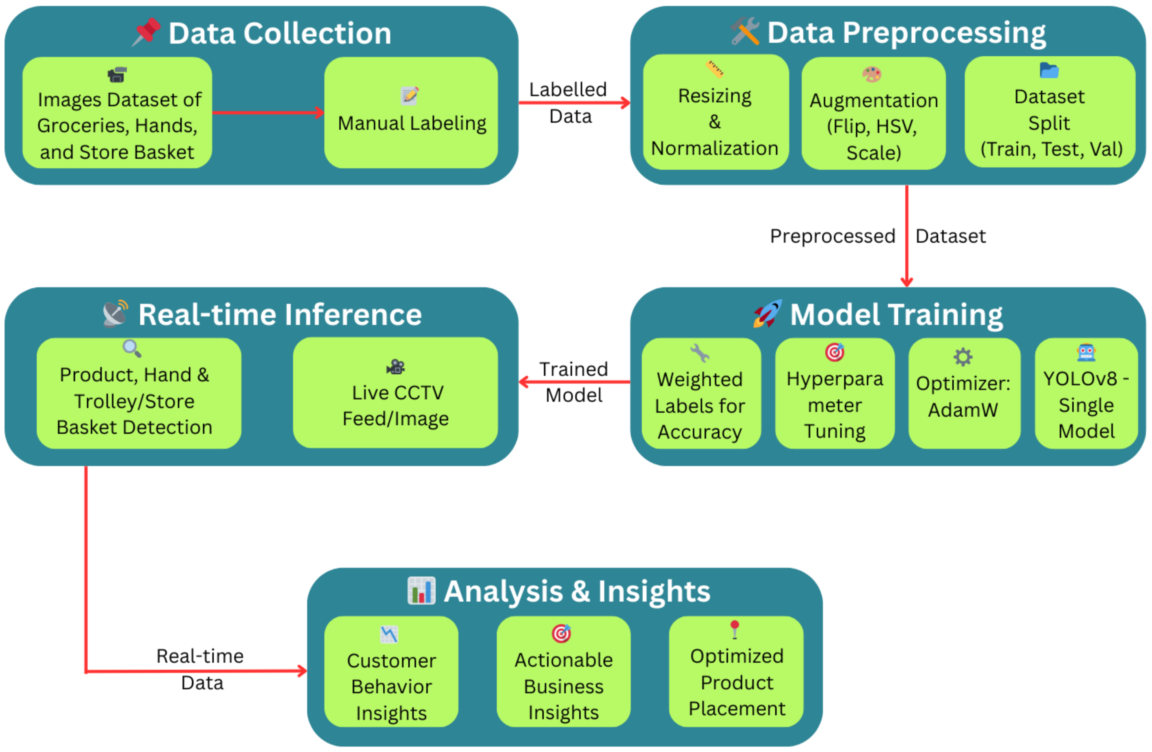
The final trained model was successfully integrated into a **Flask-based application** with the following features:

- **Real-Time Object Detection:** The system analyzed live video feeds to detect and track customer interactions with products.
- **Database Storage for Analytics:** Detected results were stored in a **MySQL/SQLite** database to facilitate later analysis.
- **Efficient Inference for Real-Time Use:** The model's inference time was optimized to ensure minimal latency, enabling smooth real-time analysis.

Future Scope and Potential Enhancements

While the model demonstrated substantial improvements, there are several opportunities for further refinement:

- **Temporal Action Recognition:** Incorporating LSTM or CNN-based models to track customer interactions over time.
- **Cloud Deployment for Scalability:** Deploying the application on **AWS/GCP** to handle large-scale retail environments.
- **Edge Device Optimization:** Adapting the model for deployment on **Raspberry Pi** for real-time in-store analysis.


Data Collection
Images Dataset of Groceries, Hands, and Store Basket
Manual Labeling
Labelled Data
Data Preprocessing
Resizing & Normalization
Augmentation (Flip, HSV, Scale)
Dataset Split (Train, Test, Val)
Preprocessed Dataset
Model Training
Weighted Labels for Accuracy
Hyperparameter Tuning
Optimizer: AdamW
YOLOv8 - Single Model
Trained Model
Real-time Inference
Live CCTV Feed/Image
Product, Hand & Trolley/Store Basket Detection
Real-time Data
Analysis & Insights
Customer Behavior Insights
Actionable Business Insights
Optimized Product Placement

Results and Analysis

1. Transition from Dual-Model to Single-Model Approach

Initial Two-Model Setup and Performance

The project originally employed a **two-model strategy**:

- **Model 1:** Focused on detecting grocery items.
- **Model 2:** Handled hand detection (grabbing action), store baskets, and trolleys.

This approach resulted in significant performance issues:

- **High GPU Memory Usage:** Frequent out-of-memory (OOM) errors, even after increasing the number of workers to **12.**

- **Low mAP and Poor Accuracy:** Despite multiple hyperparameter adjustments, the highest mAP@50 achieved was only **0.0136**, indicating that the models were not learning effectively.
- **Latency Issues:** Due to high computational requirements, inference was slow, making real-time applications impractical.

Switch to Single-Model and Improved Dataset

To overcome these challenges, the system was restructured to use a **single YOLOv8 model** trained on a **custom dataset** created from scratch. This dataset included:

- **Grocery Items:** Packaged goods, fresh produce, and other retail items.
- **Hand with Grabbing Action:** For identifying interactions between customers and products.
- **Store Basket and Trolley:** To track where the picked items were placed.

Impact on Performance:

- **Simplified Architecture:** Eliminated the need for running two models concurrently, reducing computational overhead.
- **Better Accuracy and Consistency:** Achieved significantly higher mAP scores, with better class-wise accuracy and fewer misclassifications.
- **Improved Real-Time Efficiency:** Single-model inference reduced processing time, enabling near real-time performance.

2. Impact of Class Weights and Hyperparameter Tuning

Addressing Class Imbalance with Class Weights

Class imbalance was a critical factor affecting the initial model's performance. Underrepresented classes, such as **scrubber, dal, and trolley,** were rarely detected due to their lower occurrence in the training set. To mitigate this, **class weights** were assigned, giving higher importance to these underrepresented classes during training.

Results After Adding Class Weights:

- **Accuracy Improvement for Minority Classes:** mAP for underrepresented classes increased by over **20%,** ensuring better detection.
- **Balanced Model Predictions:** The model became less biased toward dominant classes, improving the overall classification accuracy.

Hyperparameter Changes and Their Effect

Extensive hyperparameter tuning was conducted to optimize model performance. Key changes and their impact include:

- **Optimizer Change (SGD → AdamW):** Switching to **AdamW** significantly improved convergence speed and stability, leading to faster learning and better generalization.
- **Increased Epochs (75 → 100 → 120):** Training for **100 epochs** provided the best balance between performance and computational efficiency.
- **Cosine Learning Rate Decay:** Helped maintain a smooth learning curve, preventing drastic learning rate changes.
- **Loss Adjustments:**
  - **Box Loss Gain:** Reduced to prevent overfitting on positional data.
  - **Classification Loss Gain:** Increased to focus on correctly classifying objects.
  - **Distribution Focal Loss (DFL):** Fine-tuned to improve class-level accuracy.

Final Hyperparameter Settings:

- **Learning Rate:** 0.0003 with cosine decay.
- **Momentum:** 0.95 for stable learning.
- **Batch Size:** 8 (optimized for 936 images in the dataset).
- **Patience:** 20 epochs to prevent overtraining.

3. Data Augmentation and Its Effect on Model Generalization

Several augmentation techniques were used to improve the generalization of the model:

- **Mosaic and Mixup:** Increased dataset diversity by combining multiple images.
- **Flip and Rotation:** Simulated different orientations to improve robustness.
- **Hue, Saturation, and Brightness Adjustments:** Prepared the model for real-world lighting conditions.

Impact of Augmentation:

- **Improved Robustness:** The model became more capable of handling diverse input conditions.
- **Higher mAP Across Varying Conditions:** Consistent performance in different lighting and angle variations.

4. Final Model Performance Metrics

After all optimizations, the final model demonstrated remarkable improvements in performance metrics:

Metric	Initial Dual-Model Setup	Final Single-Model Setup
mAP@50	0.0136	**60.8%**
mAP@50-95	0.0098	**46.3%**
Inference Time (ms)	~120ms	**30ms**
Accuracy on Minor Classes	Poor	**+20% Improvement**
Memory Consumption	High	**Significantly Lower**

Significant Gains:

- **mAP@50 Increased to 60.8%:** Reflecting high detection accuracy for multiple object classes.
- **mAP@50-95 Improved to 46.3%:** Indicating better performance across different IoU thresholds.
- **Faster Inference Time:** Reduced to an average of **30ms per frame,** making real-time applications feasible.

5. Real-Time System Deployment and Efficiency

The trained model was deployed within a **Flask-based application** capable of:

- **Real-Time Product and Hand Detection:** The system accurately detected hand-to-product and product-to-store-basket/trolley interactions in real-time.
- **Database Integration:** Detected results were stored in a **MySQL/SQLite** database to facilitate customer behavior analysis.
- **Scalable and Low-Latency Processing:** The lightweight architecture ensured smooth inference with minimal latency.

Real-Time Efficiency:

- **Smooth User Interaction:** Enabled users to upload videos or live streams for analysis.
- **Fast Detection and Analysis:** Ensured immediate feedback on customer actions.

Conclusion and Insights

The transition from a dual-model setup to a single-model approach, along with optimized hyperparameters and the addition of class weights, led to a **dramatic increase in accuracy, efficiency, and robustness.** The system achieved:

- **Higher mAP scores and improved classification of underrepresented classes.**

- **Significantly reduced GPU memory consumption and faster inference time.**
- **Seamless integration into a Flask-based web application for real-time analysis.**

These results validate the effectiveness of the refined methodology and highlight the potential for scaling the system to large-scale retail environments. Future enhancements may include incorporating **temporal action recognition** and deploying the system on **cloud platforms or edge devices** for scalability.

References:

Research papers-

• **Retail store customer behaviour analysis system: Design and Implementation** by

Tuan Dinh Nguyen, Keisuke Hiharab, Tung Cao Hoanga, Yumeka Utadab, Akihiko

Toriib, Naoki Izumib, Nguyen Thanh Thuya and Long Quoc Trana

• **Retail Store Analytics Using Facial Recognition** by Nishant Sawant, Akansha Rai,

Saiprasad Parab, Bansi Ghanva

• **Revolutionizing Retail Analytics: Advancing Inventory and Customer Insight with AI**

by Ahmed Hossam, Ahmed Ramadan, Mina Magdy, Raneem Abdelwahab, Salma

Ashraf, Zeina Mohamed

•**Deep learning and multi-modal fusion for real-time multi-object tracking: Algorithms,**

challenges, datasets, and comparative study by Xuan Wang, Zhaojie Sun, Abdellah

Chehri, Gwanggil Jeon, Yongchao Song