ChickOut: Deep Learning–Based Poultry Retail and Inventory System with Real-Time Analytics

Fernan Frans Pelobello
Ateneo Mobile and Wireless Laboratory
Ateneo de Manila University
Quezon City, Philippines
fernan.pelobello@student.ateneo.edu

Maxine Van Caparas
Ateneo Mobile and Wireless Laboratory
Ateneo de Manila University
Quezon City, Philippines
maxine.caparas@student.ateneo.edu

Jan Kevin A. Galicia
Ateneo Mobile and Wireless Laboratory
Ateneo de Manila University
Quezon City, Philippines
fernan.pelobello@student.ateneo.edu

Wen-Yaw Chung
Department of Electronics Engineering
Chung Yuan Christian University
Taoyuan City, Taiwan
eldanny@cycu.edu.tw

Maria Leonora C. Guico
Ateneo Mobile and Wireless Laboratory
Ateneo de Manila University
Quezon City, Philippines
mguico@ateneo.edu

***Abstract*—This paper presents ChickOut, an intelligent poultry retail and inventory system that eliminates the need for barcodes by integrating deep learning–based object recognition and automated weighing for real-time checkout and inventory management. The system was developed to address inefficiencies in traditional retail operations involving non-barcoded products such as fresh chicken meat, which typically require manual logging. A custom dataset of 4,207 images (6,638 annotations) across five chicken parts—breast, leg, thigh, wing, and quarter—was used to train a YOLOv7 detection model. After 300 training epochs, the model achieved an mAP@0.5 of 97.8%, mAP@0.5:0.95 of 91.2%, precision of 96.5%, and recall of 95.8%, enabling reliable real-time detection under varying lighting and orientation. The weighing module, composed of a load cell and HX711 amplifier interfaced with an Arduino Uno, provides ±1 g accuracy and an average response time of 0.3 seconds. The Django-based backend automatically computes prices, updates stock in real time, and issues low-stock alerts, achieving 100% synchronization accuracy between sensor data and database records. An integrated web-based analytics dashboard visualizes sales trends, top-selling items, and vendor performance with <2-second data refresh latency for multiple terminals. By removing the reliance on barcodes and combining AI vision, embedded sensing, and analytics, ChickOut enhances operational efficiency, transparency, and sustainability in poultry retail—advancing UN SDGs 9 and 12.**

Keywords— retail automation, inventory management, YOLOv7, computer vision, barcode-free checkout, Django, deep learning, grocery, market, chicken, poultry

I. Introduction

Chicken remains one of the most in-demand food products due to its affordability, versatility, and nutritional value. To meet high demand, retailers aim to make purchases faster and more convenient. A promising solution is the self-checkout system for chicken and other meats, allowing shoppers to weigh, identify, and pay without queues—ensuring efficiency, freshness, and proper handling.

Integrating automated inventory with self-checkout further improves operations. Each transaction updates stock levels in real time, reducing tracking errors, avoiding shortages or overstock, and ensuring product freshness. Together, AI-assisted self-checkout and automated inventory management streamline both customer transactions and store logistics [1].

Traditional inventory systems remain manual and error-prone, often resulting in inaccuracies from spoilage, losses, or theft [2]. These lead to stockouts or misplaced items, while lack of real-time data limits responsiveness to demand [1]. Many checkout systems also fail to update inventories automatically, particularly for non-barcoded products like meat or produce [3]. Perishables further challenge accuracy, where timely tracking is vital to reduce waste.

In traditional markets such as *palengkes* in the Philippines, shared selling spaces and lack of unified systems hinder vendor-level tracking and performance monitoring. Recent advances in computer vision address these gaps by automating recognition and inventory processes [4]. AI-powered systems combined with digital weighing and backend synchronization can update inventory instantly and minimize human error [5].

This study introduces an AI-based chicken part recognition and weighing system with a Django backend that automatically updates inventory, flags low-stock items, and analyzes sales trends. It minimizes manual audits, enhances operational efficiency, and aligns with UN SDGs 9 and 12 by promoting innovation and sustainable consumption through improved management of perishable goods.

The specific objectives for the study are:

1. Create an image dataset of chicken parts and train a YOLOv7 model
2. Build retail and inventory hardware by integration of a camera and load scale for real-time varied chicken parts detection and automatic pricing.
3. Develop a dynamic web-based application featuring a real-time inventory management system with restocking alerts and inventory status classification.
4. Design a web-based analytics dashboard to visualize sales trends, identify top-selling chicken parts, and track inventory movement.

II. Review of Related Literature

A. Stocking of Items in Philippine Supermarket

In the Philippines, the locale of the study, supermarkets vary significantly in size and operations. Large chains like SM Supermarket, Puregold, and Robinsons use centralized

warehouses for multiple branches but still rely on manual reordering methods, such as visual assessments and historical sales trends. Restocking perishables like produce and proteins occurs daily or every few days due to perishability and supplier schedules [7]. Smaller supermarkets and local stores follow simpler practices, with owners conducting daily visual inspections to determine restocking needs. The sale of fresh, unbarcoded goods by weight complicates inventory tracking, leading to overstocking (resulting in spoilage) or understocking (causing missed sales and dissatisfied customers) [8], [9]. Such inefficiencies contribute to food waste and reduce customer retention [10]. Despite the benefits of real-time inventory systems, smaller retailers often view these as too costly or complex, leaving them dependent on outdated manual methods [11].

B. Retail Identification and Inventory Technologies

Recent studies highlight a shift towards integrating IoT, RFID, and AI to enhance inventory management tracking, auditing, and predictive capabilities. Zhou et al. [12] proposed an IoT-based system using RFID and blockchain to ensure transparency and tamper-proof auditing, improving asset traceability and reducing counterfeit risks. Bailkar showed that machine learning algorithms can enhance intelligent inventory management within the ABC Inventory Classification framework [13]. Klasson et al. developed a dataset for grocery environments, addressing challenges like cluttered backgrounds and inconsistent lighting in training models [14]. Convolutional Neural Networks (CNN) have also improved retail experiences, with applications in augmented reality (AR) frameworks for product identification, benefiting users with visual impairments. It emphasize how computer vision technologies enhance shopping experiences for the visually impaired. Tonioni and Stefano proposed a hierarchical embedding framework to improve item recognition accuracy under real- world conditions [18], while Roslan and Saad demonstrated the efficiency of CNNs for real-time item recognition at checkout, expediting the shopping process and reducing errors [19].

C. Dashboard for Analytics

Dashboards convert raw inventory data into actionable insights, supporting agile decision-making for store managers. While tools like D3.js and Plotly handle frontend visualization, Django is key for backend data processing and logic [7]. Our system uses Django to manage data and integrate seamlessly with D3.js, enabling visualizations of inventory movement, payment trends, order history, and item popularity [20]. These real-time dashboards, powered by Django, optimize supply chain operations and enhance the customer experience [21]. While dashboards are well-explored in fields like healthcare and smart cities, there's a gap in grocery retail, especially for managing non-barcoded inventory. Our solution fills this gap by combining Django's backend with real-time analytics and visual techniques.

III. System Flow

The system architecture of ChickOut, shown in Fig.1 an automated chicken part checkout and inventory management system, illustrates how image recognition, weight sensing, and data analytics are integrated to streamline poultry retail operations. The process begins when a user initiates a checkout, placing chicken parts on the system's weighing platform. Upon placement, both the camera and the load sensor are activated: the camera captures an image of the item and sends it to the microcontroller for object identification using a trained deep learning model, while the load sensor measures its weight. The microcontroller then computes the corresponding price based on the identified part and transmits the transaction details—including item type, weight, and price—to the connected tablet or PC.

The tablet or PC functions as the main user interface and management hub, handling remote storage, automatic stock updates, and data visualization. It also provides real-time analytics on stock performance, sales trends, and inventory levels. Users can access the system through a terminal to check out products, review data, or add new stocks, ensuring a continuous and synchronized flow of information across all system components. Once a chicken part is identified, the inventory is automatically updated in real time, ensuring accurate product tracking. Any discrepancies between stock levels and sales are immediately reflected in the database, and the system generates restocking alerts for items running low on supply.

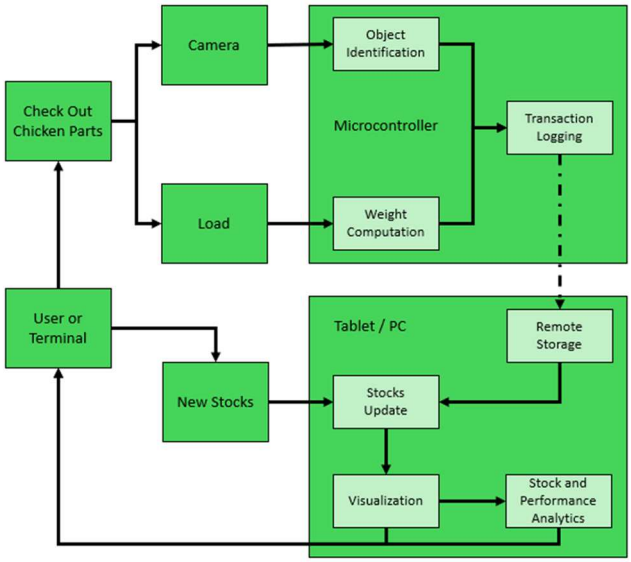


Fig. 1. Process flow of ChickOut retail and inventory system

The physical setup of the system, shown in Fig. 2, integrates the camera, load sensor, and display monitor for efficient customer interaction.

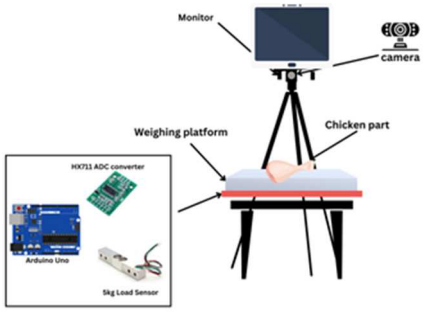


Fig. 2. Physical set up of the monitor

The system also supports multiple-item transactions, allowing several chicken parts to be detected and processed sequentially on the same weighing platform. Each time an item is placed, the system identifies the part, determines its weight, and computes the price accordingly. Customers can continue adding parts—whether of the same or different

type—and the system updates the running total in real time. This feature enables mixed combinations of chicken parts with varying prices per kilogram to be processed in a single transaction, a capability not typically found in existing retail systems. Once all items are processed and payment is completed, the inventory database is automatically updated, and the transaction is logged for record-keeping and analytics.

The real-time inventory management component plays a vital role in maintaining operational efficiency. Low-stock items are automatically flagged, triggering restocking notifications to prevent shortages. Every transaction is recorded and visualized through a web-based analytics dashboard that allows store managers to analyze sales trends, monitor inventory movement, assess product popularity, and identify both high-performing and unsold items. The dashboard also supports user-based logins, enabling individual cashier performance tracking and transaction auditing per terminal or vendor. Through the seamless integration of real-time inventory updates, multi-item transaction handling, and intelligent data visualization, ChickOut ensures accurate stock records, optimized restocking, and data-driven decision-making. This enhances transparency, operational efficiency, and customer convenience in poultry retail operations.

IV. Dataset Creation and Model Training

A custom dataset was created featuring five commonly sold chicken cuts in the Philippines: breast, leg, thigh, wing, and quarter. Images and videos were captured at varying distances and angles against different backgrounds to enhance contrast and improve model learning. Detection experiments were conducted under both controlled and realistic conditions.

For the final setup, a dark background was intentionally used to increase contrast between the chicken products and the background, allowing the chicken part to stand out. While the model is capable of operating under varied background conditions, the controlled background was selected as a practical system constraint to ensure reliable detection in real-world retail use.

The dataset consists of 4,207 images with a total of 6,638 annotations, averaging 1.6 annotations per image. The median image resolution is 1280×720 pixels, with an average size of 0.92 MP. For model development, the dataset was split into 85% training, 10% validation, and 5% testing, ensuring a balanced distribution across all classes. A visual summary of the dataset, including class distributions, is shown in Figure 2 and summarized in Table I.

TABLE I. Dataset Annotation Count

Class	Total # of Annotation	Sample Image
Breast	1443	
Leg	1353	
Thigh	1324	
Wing	1281	
Quarter Leg	1237	

YOLOv7 model was trained using a custom dataset that was preprocessed through auto-orientation and resized to 640×640 pixels. Data augmentation was applied to increase dataset variability and improve model robustness. Each original image generated four augmented versions using the following transformations:

- Rotation: −15° to +15°
- Shear: ±10° horizontally and vertically
- Brightness Adjustment: −15% to +15%
- Noise Injection: up to 0.1% of pixels

In this study, the YOLOv7 model was trained under two configurations—200 epochs and 300 epochs—to evaluate the impact of extended training on detection performance. As shown in Figure 3, the model trained for 200 epochs demonstrates stable convergence across key training metrics. Training and validation losses for bounding box regression, objectness, and classification consistently decrease, indicating effective learning and reduced prediction error. Precision and recall rise steadily and plateau at high values, signifying reliable detection performance. Similarly, mAP@0.5 and mAP@0.5:0.95 progressively improve and stabilize, confirming robust generalization and accurate object detection across various IoU thresholds.

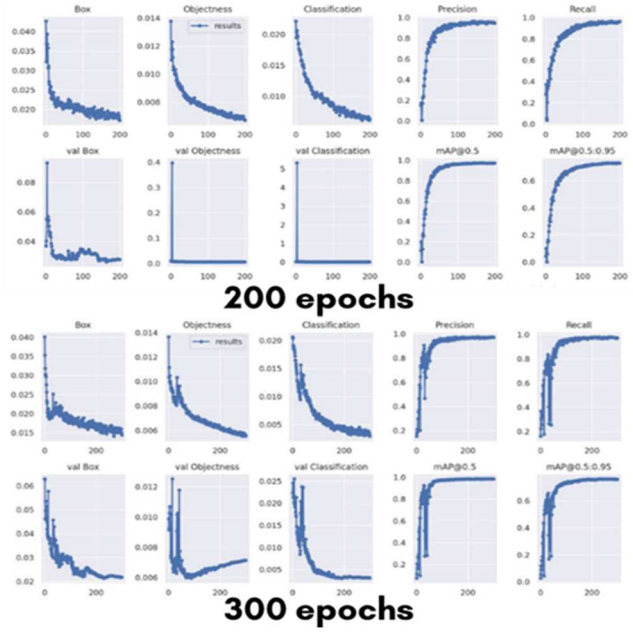


Fig. 2. YOLO Performance Metrics results

For the extended 300-epoch configuration, the model continues to refine performance, with losses exhibiting further decline and precision and recall remaining near optimal levels. The mAP metrics show slight but consistent improvement compared to the 200-epoch results, reflecting marginal gains from prolonged training. However, the improvements beyond 200 epochs are incremental, suggesting that the model has nearly converged by this point and that additional training yields diminishing returns.

V. Hardware Design and Testing

A. Hardware Setup

In building the weighing module prototype, the load cell was mounted onto a circular acrylic platform using screws and adhesive to ensure stability and prevent movement. The sensor was directly connected to the HX711 signal amplifier, which serves to convert the analog signal from the load cell into digital data readable by the microcontroller. As shown in Fig. 3, The HX711 was then interfaced with an Arduino Uno, through jumper wires, providing both signal input and power supply. The system was powered through a USB connection to a computer, allowing real-time data monitoring and calibration during testing.

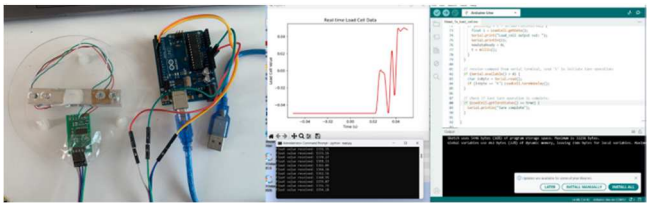


Fig. 3. Initial digital scale build and Real-time load cell monitoring.

The setup demonstrates a real-time load cell monitoring system using an Arduino Uno. Load cell data is transmitted via serial communication to a Python script, which plots the readings in real time. The plot shows load cell values fluctuating around zero, indicating detection of applied force. The Arduino code handles tare operations and continuously sends sensor values, while the Python interface receives and displays these readings dynamically.

The pySerial library was employed to facilitate data transfer from the Arduino to Python. This library enables serial communication between the two platforms by transmitting data through serial ports. To ensure proper synchronization, the baud rate used in the Arduino was matched with the one specified in the Python script for weight readings — in this case, 115200. Since the Arduino sends purely numerical data, Python decodes the byte stream using UTF-8, converting it into floating-point values that represent the weights.

The goal is not only to read the current weight measurements but also to identify significant weights — each corresponding to an item placed on the scale. Once a significant weight is detected, it is stored in the items_weights list, where the total number of elements represents the number of items currently on the scale. These weights must dynamically update in real time, adding or removing entries to accurately reflect the items on the scale.

To achieve this, several key parameters were defined in the Python weight-reading code:

- Zero Threshold – Ensures that the list stays empty when near-zero readings are detected. Any reading below a minimal threshold (1 gram) signifies an empty scale and resets the item list.
- Stability Threshold – Determines if weight readings are stable by checking whether the change between consecutive readings is within a small range (0.1 gram). If so, the reading is considered stable, indicating a new item has likely been placed.
- Stability Count – Defines the number of consecutive identical readings (set to 3) required for a weight to be recognized as significant.
- Minimum Item Weight – Filters out insignificant variations such as accidental touches or vibrations. Only stable weights exceeding 4 grams are considered valid and added to the list.
- Weight Tolerance – Used when an item is removed. The system searches for a previously recorded item whose weight closely matches (within ±2 grams) the observed negative weight change, then removes it from the list.

Fig. 4 shows the final complete setup of ChickOut including the weighing platform, a mounted the register and inventory dashboard. A black tray was screwed onto the top block of wood which would serve as the surface for weighing chicken parrts. The tray's color was chosen to ensure sufficient contrast against light-colored items, matching dark backgrounds seen in the dataset.

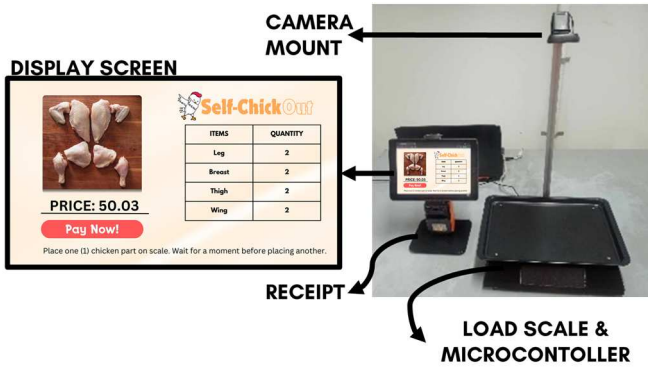


Fig. 4. Hardware set up of ChickOut

B. Accuracy and Speed of Weight Estimation

To evaluate the accuracy of the weight sensor, several trials were conducted using different chicken parts, and the measured weights were compared with their actual values. Table VIII presents the results of the comparison. The computed percentage differences across all samples ranged from 0.007% to 0.17%, with an overall average deviation of only 0.069%.

The smallest deviation was recorded for the chicken leg (0.007%), while the largest was observed for the chicken thigh (0.17%). These minor variations may be attributed to slight differences in placement on the weighing platform or surface irregularities of the samples. Despite these minimal discrepancies, all readings were well within a 0.2% margin of error, indicating that the sensor consistently delivered precise and reliable measurements.

Overall, the results demonstrate that the load sensor exhibits excellent linearity and repeatability when used for

poultry products. The high degree of accuracy confirms the system's capability to provide dependable weight readings necessary for automated checkout and inventory applications.

TABLE II. DATASET ANNOTATION COUNT

Chicken Part	Actual Weight (kg)	Sensor Weight (kg)	Percentage Difference
Breast	0.1799	0.17976	**0.078%**
Leg	0.1517	0.15169	**0.007%**
Thigh	0.1645	0.16422	**0.170%**
Wing	0.0991	0.09911	**0.010%**
Quarter	0.3056	0.30584	**0.078%**

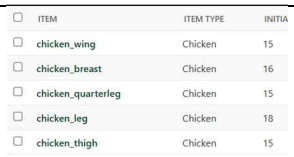
To evaluate the system's response time, several trials were performed using different chicken parts. The duration from the moment an item was placed on the scale until its corresponding weight was registered in the Python list was measured. Similarly, the time it took for the scale to return to zero after the item's removal was recorded. The system required an average of 0.3 sec to stabilize and record a valid weight, and approximately 0.3 to reset to zero. These response times indicate that the weighing module operates efficiently and is well-suited for real-time processing in automated checkout applications.

VI. REAL-TIME INVENTORY IMPLEMENTATION

For the inventory management component of this system, we utilized the built-in admin dashboard and SQLite database provided by the Django framework. We customized it by enhancing its visual design and integrating graphs to display key inventory metrics, such as current stock levels, sales trends, and demand patterns. The homepage of the admin dashboard prominently features a sales trend chart. This line graph illustrates daily sales activity over the course of a week. Users can also apply filters to visualize sales trends over a specific time period. Below the sales trend chart, the dashboard displays the core data models of the system, namely: items_price, Order Transaction, and Stock.

TABLE III. DIFFERENT PAGES OF ADAPTIVE INVENTORY SYSTEM

Model or Page	Description
Item Price Model  ITEM NAME / ITEM PRICE chicken_wing 39.0 chicken_breast 45.0 chicken_quarterleg 74.0 chicken_thigh 60.0 chicken_leg 55.0	▪ stores the price per kilogram of each chicken part (used to compute the total price) ▪ records the most recent date and time an item's price was changed.
Order Transaction Model  ORDER ID / TOTAL AMOUNT / PAYMENT MODE T202505080008 48.17 Online T202505080007 74.08 Cash T202505080006 53.28 Cash T202505080005 52.65 Cash T202505080004 81.12 Cash T202505080003 12.80 Cash T202505080002 82.53 Cash	▪ visualizes customer orders, including unique order ID constructed (using date-time and cash register), the total amount, payment method, and the date and time of processing.
Current Stock Model  Stocks Chart chicken_breast chicken_leg chicken_quarterleg chicken_thigh chicken_wing	▪ monitors both the initial and remaining stock of each item after transactions, along with the last update timestamp. ▪ displays the stock status a. "Good Stock" if the remaining quantity is above threshold value (e.g. 50%) b. "Low Stock" if it is below threshold. c. "Out of Stock" if the stock reaches zero. • operators can adjust this threshold based on their operational policies, product turnover rates, or restocking strategies.


ITEM / ITEM TYPE / INITIA
chicken_wing Chicken 15
chicken_breast Chicken 16
chicken_quarterleg Chicken 15
chicken_leg Chicken 18
chicken_thigh Chicken 15

VII. WEB-BASED DATA ANALYTICS IMPLEMENTATION

To effectively build a web-based analytics dashboard, we leveraged two Django ORM models—history stock and expected dryout—that serve complementary roles in inventory and sales analytics. Overall, these models facilitate the development of an analytics dashboard by exposing structured and query-able data fields that enable deeper insights into inventory performance.

TABLE IV. STOCKS AND PERFORMANCE ANALYTICS

Model or Page	Description
History of Batch Model  Orange Batch Overview BATCH / BATCH DATE / DISPOSAL DATE / TOTAL QTY / SALES QTY / REMA 3 April 22, 2025, 6 p.m. April 29, 2025, 6 p.m. 50 13 37 2 April 8, 2025, 6 p.m. April 15, 2025, 6 p.m. 50 45 0 1 April 1, 2025, 6 p.m. April 8, 2025, 6 p.m. 53 53 0	▪ includes fields for total stock, sales, remaining items, perishables, and losses. ▪ offers granular visibility into how each batch of items is consumed or wasted over time. ▪ supports historical tracking of sales trends, stock depletion, and wastage patterns of each batch delivery
Expected Dryout Model  ITEM / CURRENT STOCK / AVERAGE SALES / EXPECTED DRYOUT DATE Chicken_breast 3 5 Oct. 28, 2025, 6 a.m. Chicken_thigh 6 4 Oct. 28, 2025, noon Chicken_quarterleg 10 3 Oct. 30, 2025, noon Chicken_wing 4 3 Oct. 30, 2025, 6 a.m. Chicken_leg 16 2 Oct. 30, 2025, 4:11 p.m.	▪ performs predictive analytics by computing how long the current stock will last based on past sales velocity. ▪ calculates the duration from delivery to present day, computes average sales per day, and estimates the expected dryout date.
Terminal Performance Model  Select cashier_performance to change Cashier Performance Chart cashier_name Cashier1 Cashier2 Cashier3	▪ designed for managers and owners to effectively analyze the performance of individual cashiers, checkout stations, and sellers.

Top-selling items can be identified using the sales_qty field from the history stock model, complemented by average_sales data from the expected dryout model. Meanwhile, unsold or underperforming batches can be determined by analyzing entries with high remaining_qty and either low sales_qty or low average_sales, indicating poor movement or excess stock.

Additionally, tracking inventory movement by category or batch is made possible through filters applied to fields such as batch and date_batch, as well as computed values like duration, expected_dryout_date, and perished_qty. These data

points collectively support dynamic visualizations and reports for sales trend analysis and inventory management.

From testing, the developed inventory system can successfully register individual cashier logins, associate's transactions with specific user accounts, and links those with designated terminals or checkout stations. The backend consistently updates transaction logs in real time, and the dashboard accurately reflects sales activity based on the logged-in user and terminal ID. The preliminary dashboard views confirm that store managers will be able to visualize and compare seller or terminal performance through bar graphs, pie charts, and filtered reports.

VIII. CONCLUSION

This paper presented ChickOut, an innovative deep learning–based poultry retail and inventory management system that automates the identification, weighing, and recording of chicken parts in real time—without the need for barcodes or manual input. The integration of computer vision and load-sensing technologies enables efficient and accurate transaction processing suitable for modern retail environments.

The vision module, powered by a YOLO-based detection model, achieved a high recognition accuracy of 97.8% mAP@0.5, effectively identifying various chicken parts even when multiple and different parts are placed simultaneously on the platform. This multi-item detection capability distinguishes ChickOut from traditional single-item systems, allowing faster processing and improved customer throughput.

The weighing module demonstrated an average response time of 0.3 seconds, ensuring rapid stabilization and real-time computation of combined weights and prices. This responsiveness makes the system ideal for continuous and high-traffic retail operations.

Through its synchronized database and dashboard integration, ChickOut maintains accurate, real-time inventory updates by automatically logging each transaction's visual, weight, and pricing data. This ensures precise stock tracking, reduces manual errors, and provides vendors with actionable analytics for sales and supply management.

Overall, ChickOut introduces a novel and practical approach to automating retail transactions in the poultry industry—combining multi-item detection, intelligent weighing, and seamless inventory synchronization. Future work will focus on expanding dataset diversity, enhancing robustness under varying market conditions, and incorporating mobile payment integration. With its unique combination of speed, accuracy, and analytics, ChickOut represents a significant step toward smarter, data-driven, and fully automated retail systems.

REFERENCES

[1] U. Orumie and F. Nzerem, "Inventory control system on some products of some selected supermarkets in port-harcourt rivers state", Asian Journal of Mathematics and Computer Research, p. 1-16, 2023.

[2] K. Aswathy, A. Shaji, M. Athul, & K. Bhavya, "Smart inventory management system", Interantional Journal of Scientific Research in Engineering and Management, vol. 09, no. 04, p. 1-9, 2025.

[3] A. Deshpande, I. Deshpande, A. Kulkarni, & K. Shinde, "A survey on purchase and inventory management systems using barcode mechanism", International Research Journal of Modernization in Engineering Technology and Science, 2023.

[4] S. Patel, "Multi-modal product recognition in retail environments: enhancing accuracy through integrated vision and ocr approaches", World Journal of Advanced Research and Reviews, vol. 25, no. 1, p. 1837-1844, 2025.

[5] M. Yakubu and Y. Abubakar, "Assessment of the efficiency of customer order management system: a case study of sambajo general enterprises jigawa state, nigeria", International Journal of Computer Applications Technology and Research, p. 306-311, 2023.

[6] C. Abhinaya, B. Lahari, C. Priya, D. Anjali, B. Navya, & B. Jyothi, "Supermarket sales prediction using machine learning", Epra International Journal of Research & Development (Ijrd), p. 1-6, 2023.

[7] P. Pascual, N. Sedanza, M. Loso, M. Salvino, R. Mendoza, & N. Abenis, "Understanding consumer buying behaviours towards public markets and grocery stores in tacloban city, philippines", International Journal of Engineering Technologies and Management Research, vol. 6, no. 3, p. 40-47, 2020.

[8] M. Ghosh - Dastidar, G. Hunter, R. Collins, S. Zenk, S. Cummins, R. Beckmanet al., "Does opening a supermarket in a food desert change the food environment?", Health & Place, vol. 46, p. 249-256, 2017.

[9] E. Watanabe, C. Nascimento, M. Freitas, & M. Viana, "Food waste: an exploratory investigation of causes, practices and consequences perceived by brazilian supermarkets and restaurants", British Food Journal, vol. 124, no. 3, p. 1022-1045, 2021.

[10] L. Fotabong and A. Baliukevičius, "The influence of supply chain management on the organizational performance of supermarket",, 2024.

[11] T. Olarewaju, S. Dani, C. Obeng-Fosu, T. Olarewaju, & A. Jabbar, "The impact of climate action on the financial performance of food, grocery, and supermarket retailers in the uk", Sustainability, vol. 16, no. 5, p. 1785, 2024

[12] Y. Zhou, J. Liu, and X. Zhao, "IoT-based inventory tracking system using RFID and blockchain for transparency and tamper-proof auditing," IEEE Transactions on Industrial Informatics, 2021.

[13] S. Bailkar, K. Shenoy, A. Bedekar, S. Bankar and P. More, "Smart Inventory Optimization using Machine Learning Algorithms," 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2024, pp. 1395-1400

[14] M. Klasson, C. Zhang, & H. Kjellström, "Using variational multi- view learning for classification of grocery items", SSRN Electronic Journal, 2020.

[15] M. Klasson, C. Zhang, & H. Kjellström, "A hierarchical grocery store image dataset with visual and semantic labels",, p. 491-500, 2019.

[16] Y. Alhamdan, "Leveraging from artificial intelligence (ai) in advancing the augmented reality (ar) grocery shopping experience",, 2023.

[17] J. Grashuis, T. Skevas, & M. Segovia, "Grocery shopping preferences during the covid-19 pandemic", Sustainability, vol. 12, no. 13, p. 5369, 2020.

[18] A. Tonioni and L. Stefano, "Domain invariant hierarchical embedding for grocery products recognition", Computer Vision and Image Understanding, vol. 182, p. 81-92, 2019.

[19] N. Roslan and A. Saad, "Point of sale system using convolutional neural network for image recognition in grocery store", International Journal of Artificial Intelligence, vol. 10, no. 2, p. 79-92, 2023.

[20] M. Ghosh - Dastidar, G. Hunter, R. Collins, S. Zenk, S. Cummins, R. Beckmanet al., "Does opening a supermarket in a food desert change the food environment?", Health & Place, vol. 46, p. 249-256, 2017.

[21] E. Watanabe, C. Nascimento, M. Freitas, & M. Viana, "Food waste: an exploratory investigation of causes, practices and consequences perceived by brazilian supermarkets and restaurants", British Food Journal, vol. 124, no. 3, p. 1022-1045, 2021