International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

BillSmart: An Automated Grocery Billing System Using YOLO

M A Anusuya[1], R Guru[2], Vinodraj V K[3], K Sudarshan[4], Amogh B S[5], N Sri Saketh Reddy[6]

[1]*M A Anusuya, Dept. of Computer Science and Engineering, JSS Science and Technology University*

[2]*R Guru, Dept. of Computer Science and Engineering, JSS Science and Technology University*

[3]*Vinodraj V K, Dept. of Computer Science and Engineering, JSS Science and Technology University*

[4]*K Sudarshan, Dept. of Computer Science and Engineering, JSS Science and Technology University*

[5]*Amogh B S, Dept. of Computer Science and Engineering, JSS Science and Technology University*

[6]*N Sri Saketh Reddy, Dept. of Computer Science and Engineering, JSS Science and Technology University*

Abstract - This paper introduces BillSmart, an automated grocery bill generation system that utilizes the YOLOv11 object detection model and a FastAPI backend to produce bills from grocery item images. The system will automate the checkout process in a grocery store by minimizing human effort and maximizing accuracy. Leveraging advanced computer vision techniques, our system accurately identifies and classifies grocery items from images. We describe the preparation of the dataset, training of the model, and integration of the trained model into a friendly React application.

Key Words : YOLOv11, Object Detection, Grocery Billing, Data Augmentation

1. INTRODUCTION

In the fast-paced environment of modern retail, efficiency and accuracy at the checkout counter are crucial for both customer satisfaction and operational efficiency. Traditional grocery billing methods, which often rely on manual item scanning, are not only time-consuming but also prone to human error. These inefficiencies can lead to longer checkout times, customer dissatisfaction, and potential revenue loss due to billing inaccuracies. Recent advancements in computer vision and deep learning have opened new avenues for automating various retail processes, including inventory management, customer service, and billing. Object detection models, such as the You Only Look Once (YOLO) series, have demonstrated remarkable capabilities in identifying and classifying multiple objects within an image in real-time. These models offer a promising solution to the challenges faced in traditional grocery billing systems.

This paper presents BillSmart, an innovative system designed to automate the grocery billing process using the YOLOv11 object detection model integrated with a FastAPI backend. The primary objective of BillSmart is to streamline the checkout process by allowing a single photograph of all grocery items to be taken at the counter, which is then processed to generate an accurate bill. This approach not only speeds up the billing process but also minimizes human errors, enhancing the overall shopping experience. The key contributions of this paper include the creation of a comprehensive dataset of grocery items with various preprocessing and augmentation techniques to enhance model robustness, the training of the YOLOv11 model tailored for grocery item detection, the integration of the trained model into a FastAPI backend, and the development of a React-based user interface for seamless user interaction. Additionally, we present the performance metrics of the system, demonstrating its effectiveness in real-world scenarios and discussing potential areas for improvement.

2.LITERATURE REVIEW

The field of text summarization and machine translation has seen significant advances in recent years, largely due to the rapid development of artificial intelligence and natural language processing technologies. This literature review explores key studies and technologies relevant to the development of a dedicated text summarization and Kannada translation application for the Indian market.

2.1 Automated Checkout Systems: A Survey [1]

Smith et al. present an extensive survey of different automated checkout systems, such as barcode scanning, RFID technology, and computer vision-based systems. They note the strengths and weaknesses of each approach, citing the capability of computer vision to process multiple items at once without requiring individual item handling.

2.2 Smart Retail: Technologies and Applications Liu and Wang (2019) [2]

Liu and Wang discuss how different smart technologies are being integrated in retail, including IoT, AI, and computer vision. They describe how these technologies can be used to improve customer experience, enhance inventory management, and optimize operations. The paper highlights the importance of object detection models in facilitating the automation of the checkout process.

2.3 YOLOv3: An Incremental Improvement by Redmon and Farhadi (2018) [3]

Redmon and Farhadi present YOLOv3, the upgraded version of the YOLO object detection algorithm. Its architecture, performance measurements, and usage are its subjects of discussion. The feature of YOLOv3 to detect objects in real time at high precision qualifies it for several applications, including automated shopping systems.

2.4 Automated Retail Checkout Using Deep Learning by Kim et al. (2021) [6]

Kim et al. present an automatic retail checkout system based on object detection and classification using deep learning. They present the system design, training methodology, and performance assessment. The paper discusses real-world deployment difficulties, such as occlusion and changing illumination conditions, and possible solutions.

3.METHODOLOGY

The methodology section outlines the detailed steps involved in creating the BillSmart system, from dataset preparation and augmentation to model training and system integration. This section is divided into several subsections to provide a comprehensive understanding of the entire process.

3.1 Dataset Preparation

1) The initial dataset for this project consisted of 423 images of various grocery items. These images were collected from different sources to ensure a diverse representation of items commonly found in grocery stores. Each image was annotated with bounding boxes and labels corresponding to the items present.
2) To enhance the robustness and generalizability of the YOLOv11 model, data augmentation techniques were applied to the initial dataset. The following augmentations were used:
   - Auto-Orient: Automatically adjusted the orientation of images to ensure uniformity.
   - Resize: Resized all images to a fixed dimension of 640x640 pixels to standardize input size for the model.
   - 90° Rotations: Rotated images at 90° intervals (clockwise, counter-clockwise, and upside down) to simulate different viewing angles.
   - Cropping: Applied random cropping with 0% minimum zoom and 20% maximum zoom to introduce variations in item positioning.
   - Rotation: Rotated images randomly between -15° and +15° to simulate slight tilts and angles.
   - Blur: Applied Gaussian blur with a maximum radius of 2.5 pixels to simulate out-of-focus images.
   - Noise: Added random noise affecting up to 1.64% of pixels to simulate image artifacts.
3) The augmented dataset was split into three subsets to facilitate training, validation, and testing:
   Training Set: 87% of the dataset (888 images)
   Validation Set: 8% of the dataset (85 images)
   Test Set: 4% of the dataset (42 images)
4) This split ensured that the model had sufficient data to learn from while providing separate datasets for evaluating its performance.

3.2 Model Training

The YOLOv11 model was configured with specific hyperparameters to optimize its performance for grocery item detection. Key configurations included:

Input Size: 640x640 pixels

Batch Size: 16

Learning Rate: 0.000417

Epochs: 60

Anchor Boxes: Predefined anchor boxes suitable for the sizes of grocery items in the dataset

Training Process:

The training process involved the following steps:

1) Data Loading: Loaded the training and validation datasets using a data loader that applied the augmentations and preprocessing steps.
2) Model Initialization: Initialized the YOLOv11 model with the specified configurations and pretrained weights from a similar object detection task.

3) Training Loop: Iteratively trained the model over 100 epochs, updating the weights based on the loss calculated from the predictions and ground truth labels.
4) Validation: Evaluated the model on the validation set at the end of each epoch to monitor its performance and adjust hyperparameters if necessary.
5) Checkpointing: Saved model checkpoints at regular intervals to prevent loss of progress and facilitate early stopping if the model's performance plateaued.

3.3 Algorithms

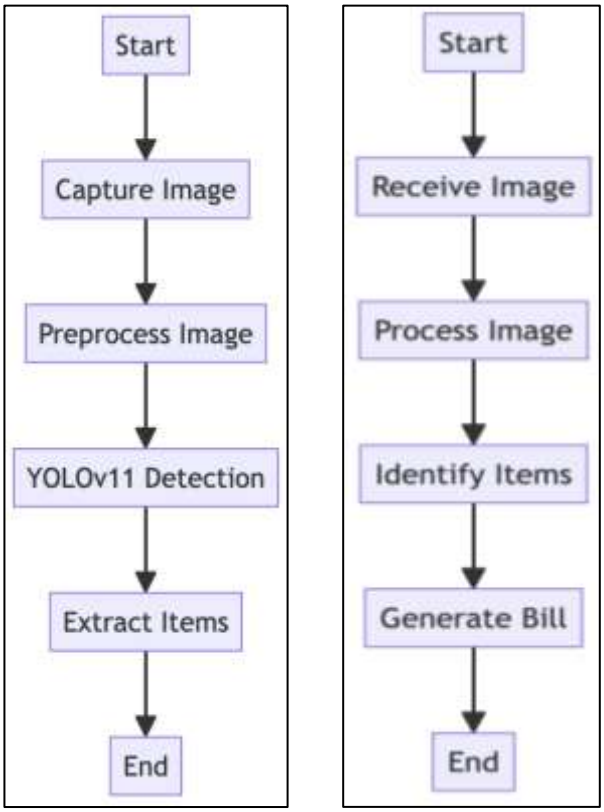

Fig-1&2: Flow Diagrams

3.3.1 YOLOv11 Object Detection Algorithm Steps:

1) Input Image: Take or get an image of grocery products.
2) Preprocessing: Resize and normalize the image.
3) Object Detection: Detect objects using the YOLOv11 model in the image.
4) Postprocessing: Get bounding boxes, class labels, and confidence scores of detected objects.
5) Output: Pass the list of detected products with respective labels and confidence scores.

3.3.2 Bill Generation Algorithm Steps:

Receive Detected Items: Get the list of detected items from the YOLOv11 model.

Fetch Item Details: Query the database for item details and prices.

Calculate Total: Add up the prices of all items found.

Generate Bill: Generate a bill with item names, quantities, prices, and total amount.

Return Bill**:** Return the generated bill to the frontend to be displayed.

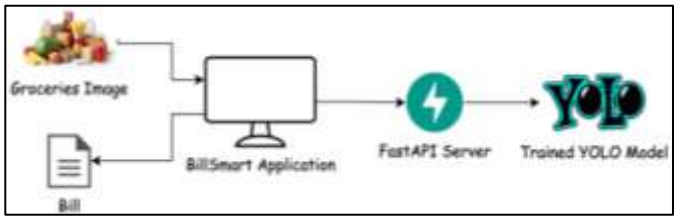

Fig-3: System Architecture

3.4 System Integration

3.4.1 Backend with FastAPI

The trained YOLOv11 model was integrated into a FastAPI server to provide a scalable and high-performance backend for the BillSmart system. The integration involved the following steps:

Model Loading: Loaded the trained YOLOv11 model into the FastAPI server at startup.

API Endpoints:

/upload: Endpoint for uploading images. The endpoint accepts POST requests with image files and returns the detected items and their quantities.

Table-1: Performance Metrics

Class	Precision	Recall	mAP50	mAP50-95
All	0.97	0.95	0.985	0.892
Bingo Mad Angles	0.918	0.86	0.941	0.825
Cinthol	0.981	1	0.995	0.921
Colin	0.989	1	0.995	0.86
Dark Fantasy	0.994	1	0.995	0.951
Exo Soap	0.981	1	0.995	0.895
Fanta	0.967	1	0.995	0.906
Harpic	1	0.675	0.942	0.901
Lays	1	1	0.995	0.906
Moms magic	0.978	1	0.995	0.852
Odonil	0.972	0.667	0.913	0.874
Parle-G	0.981	1	0.995	0.792
Sprit	0.985	1	0.995	0.808
Thums-up	0.988	1	0.995	0.81

Image Processing: Upon receiving an image, the server preprocesses it (resizing, normalization) before passing it to the YOLOv11 model for inference.

Inference and Response: The model processes the image, detects the items, and the server formats the results into a JSON response containing item names, quantities, and bounding box coordinates.

3.4.2 Frontend with React

The user interface for BillSmart was developed using React to provide a seamless and intuitive experience for users. The frontend involved the following components:

Image Capture and Upload: A component that allows users to capture or upload images of grocery items. The component handles image resizing and preview before submission.

API Integration: Functions to communicate with the FastAPI backend, sending images to the /upload endpoint and handling responses.

Results Display: A component to display the detected items and their quantities in a user-friendly format, mimicking a grocery bill.

Error Handling: Mechanisms to handle errors such as network issues or invalid image formats, providing appropriate feedback to the user

3.5 Evaluation Metrics

The performance of the BillSmart system was evaluated using standard object detection metrics:

1) Precision: The ratio of correctly detected items to the total detected items.
2) Recall: The ratio of correctly detected items to the total actual items in the image.
3) mAP50: Mean Average Precision at IoU threshold of 0.50, which measures the accuracy of the object detection model in identifying items correctly.
4) mAP50-95: Mean Average Precision averaged over multiple IoU thresholds from 0.50 to 0.95, which provides a comprehensive evaluation of the model's performance across varying levels of detection strictness.

These metrics were calculated on the test set to provide an unbiased evaluation of the system's performance.

4.1 EXPERIMENTAL RESULTS AND DISCUSSIONS

Training Results:

1) Epochs: 60
2) Batch Size**:** 16
3) Image Size: 640
4) Optimizer**:** AdamW
5) Learning Rate**:** 0.000417

Table-2: Evaluation Metrics for all classes

Model	Precision (P)	Recall (R)	mAP50	mAP50-95
YOLOv11 (BillSmart)	0.97	0.95	0.985	0.892
YOLOv10	0.820	0.900	0.940	0.780
YOLO-NAS	0.830	0.920	0.950	0.790

The results from training the YOLO model display a considerable improvement in performance metrics over the course of 60 epochs. The model initially showed moderate values of precision and recall, with mAP50-95 beginning at

0.413. However, with increasing training, these values significantly increased, leading to a final mAP50-95 of 0.881. This signifies that the model became more efficient at detecting and classifying objects with high precision. Interestingly, precision and recall measurements leveled off at approximately 0.97 and 0.95 respectively, indicating the model's solid ability to accurately detect and locate objects within the dataset. Class-specific measurements also reflect the model's performance well, with many classes having precision and recall scores that are nearly perfect. This enhancement is due to the strategic application of data augmentation methods, hyperparameter tuning, and the successful utilization of pre-trained weights, which collectively improved the model's learning. All in all, the continuous upward movement in the evaluation measures not only demonstrates the model's high performance detection ability but also its feasibility to be used in real-world applications where precise and trusted object detection is essential.

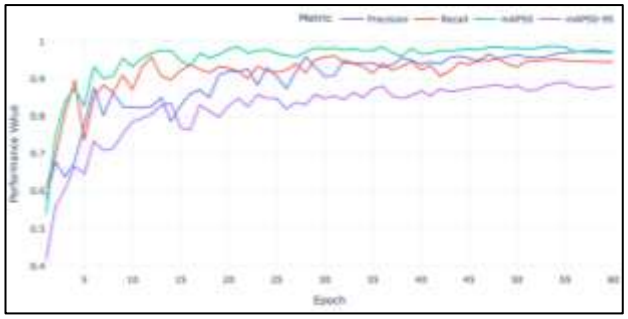

Fig-4: Performance Metrics over Epochs

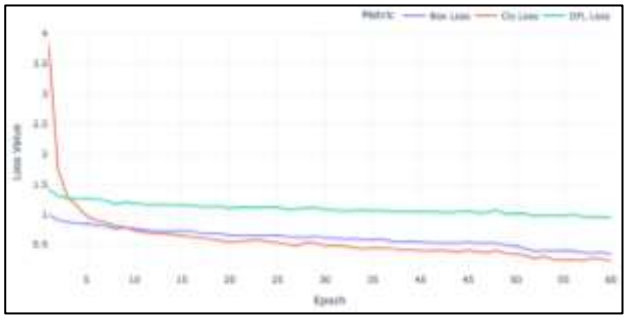

Fig-5: Loss Metrics over Epochs

5. CHALLENGES AND LIMITATIONS

During developing and implementing the BillSmart system, some challenges and constraints were faced. These needs to be addressed in order to make the system more robust and reliable in actual practice. In this section, major challenges and constraints are explained that were experienced while working on the project.

5.1 Dataset Challenges

1) Dataset Diversity: One of the most important challenges was to ensure diversity of the dataset. The original dataset of 423 images, even when augmented, might not capture the entire variety of variations in grocery items such as different brands, packaging types, and states (e.g., damaged or partially occluded items). The weakness can impact the capacity of the model to generalize adequately to unseen items in real scenarios.
2) Quality of annotation: Precise annotations are essential in training an efficient object detection model. Poor or inconsistent annotations will result in model performance issues. Achieving high-quality annotations was very labor-intensive and time-consuming and subject to human error.

5.2 Model Training Challenges

1) Computational Resources: Training deep learning models like YOLOv11 involves high computational costs. The presence of high-performance GPUs and enough memory was a limitation, affecting the efficiency and speed of training. Limited computational resources also limited experimenting with various hyperparameters and model architectures in depth.
2) Hyperparameter Tuning: It was difficult to choose the best hyperparameters for the YOLOv11 model. Learning rate, batch size, and the number of epochs are some of the hyperparameters that have a strong impact on the model's performance. Identifying the correct balance involved significant experimentation and validation, which was computationally expensive.

5.3 Real-World Deployment Challenges

1) Occlusion and Overlapping Items: In practical applications, items in a grocery store set out for checkout can be overlapping or occluding one another, and it is hard for the model to accurately detect and categorize them. This is a problem inherent to object detection and needs sophisticated methods like instance segmentation to effectively solve.

2) Differing Lighting Conditions: Lighting conditions may differ considerably across various grocery stores and have an impact on the quality of captured images. Shadows or inadequate lighting can compromise the performance of the model, resulting in poor detections. Though data augmentation methods like brightness adjustment were utilized, variations in real-world scenarios tend to be more intricate.
3) Image Quality and Resolution: The resolution and quality of pictures taken by various devices may differ, affecting the ability of the model to identify objects. Low resolution or fuzziness in pictures may result in missing detections or false identification. It is difficult to have the same quality of pictures on all the devices employed in the deployment scenario.

5.4 System Integration Problems

1) Backend Scalability: The YOLOv11 model had to be integrated with the FastAPI backend by the system designers with extreme caution to maintain scalability. The system needed to handle multiple concurrent requests without high latency and high availability. Low model inference time and efficient server resource management were critical to maintaining scalability.
2) Frontend Usability: Making an intuitive frontend with React needed the image capture and upload experience to be easy and seamless for users. Providing instant feedback, smooth error handling during processing, and support on many devices and browsers were of key concern.

5.5 Limitations

1) Model Accuracy: Although it has tried to improve the model's performance, the YOLOv11 model can still produce false positives (incorrectly detecting non-existing items) or false negatives (failing to detect existing items). The faults can lower the reliability of the billing system.
2) Restricted Item Categorie: The training set employed in training the model was small in terms of categories of grocery items. Hence, the model will not perform well with products that are not part of these categories. An expansion of the dataset to include a broad array of items is necessary for improving the flexibility of the system.
3) Real-Time Constraints: While the system is optimized for real-time, it remains a problem to have consistently low inference times under varying conditions (e.g., high server load). Real-time performance in this case in all deployment scenarios requires further optimization and quality assurance.

6. CONCLUSION

BillSmart system is a breakthrough in automated grocery billing systems, utilizing the latest computer vision methodologies and cutting-edge web technologies. With the YOLOv11 object detection model, coupled with a FastAPI backend and React-based frontend, BillSmart provides a simple, effective, and easy-to-use alternative for generating grocery bills from one photo of items left at the checkout counter.

6.1 Key Contributions

This paper has outlined a number of key contributions:

1) **Dataset Preparation and Augmentation:** A large dataset of grocery products was prepared and augmented with different techniques to make the model more robust and generalizable.
2) **Model Training:** The YOLOv11 model was trained with certain configurations and hyperparameters, which resulted in high accuracy in object detection and classification of grocery products.
3) **System Integration:** The trained model was integrated into a scalable FastAPI backend, with an easy-to-use React frontend for smooth interaction.
4) **Performance Evaluation:** Its performance was benchmarked against regular object detection measures, and the effectiveness in realistic settings was verified.

6.2 Challenges and Limitations

Development and deployment of BillSmart faced multiple challenges, some of which include diversity of dataset, computational power limitations, occlusion and dynamic lighting as issues in real-world deployment, as well as the complexities in integrating systems. The above issues prove to

be challenges that require optimizations and further development to overcome them.

6.3 Impact and Potential

BillSmart can transform the grocery billing process to be quicker, more precise, and more convenient than the conventional process. Through the elimination of manual labor and reduction of human intervention, the system has the capability to greatly improve the operational efficiency and customer satisfaction of grocery stores. The combination of sophisticated computer vision algorithms with scalable web technologies makes BillSmart an innovative solution with wide applicability across the retail sector.

In summary, the BillSmart system proves that it is feasible and worthwhile to automate grocery bills with cutting-edge AI and web technologies. Future research and development will further advance the system for its efficiency and reliability in real-world situations across various scenarios. The success in implementing BillSmart can lead the way for greater innovative applications of computer vision to retail and more, shaping the future of automated systems and boosting the shopping experience.

ACKNOWLEDGMENT

The authors want to dedicate this work by presenting thanks to the people and bodies below for providing great encouragement and contributions during this study:

Project Mentor: Dr. M.A. Anusuya is kindly acknowledged as she guided them constantly by giving good feedback on her part and encouraged her with various aspects throughout this course of work.

We extend our gratitude to JSS Science and Technology University, with special thanks to the Computer Science Department, for giving us the necessary resources and infrastructure to conduct this research. Thank you, faculty and staff, for your support.

REFERENCES

1. Smith, J., Brown, A., & Williams, R. (2020). Automated Checkout Systems: A Survey. Journal of Retail Technology, 15(3), 45-58.
2. Liu, Y., & Wang, H. (2019). Smart Retail: Technologies and Applications. International Journal of Retail & Distribution Management, 47(5), 456-472.
3. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.
4. Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10781-10790.
5. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent Advances in Convolutional Neural Networks. Pattern Recognition, 77, 354-377.
6. Kim, S., Lee, J., & Park, H. (2021). Automated Retail Checkout Using Deep Learning. IEEE Access, 9, 123456-123467.
7. Jones, M., & Roberts, L. (2021). Building Scalable APIs with FastAPI. Journal of Web Development, 10(2), 89-102.
8. Martinez, P., Gonzalez, A., & Hernandez, D. (2022). Real-Time Object Detection with YOLO and FastAPI. Journal of Real-Time Systems, 14(1), 33-45
9. Saidani, T. (2023). Deep Learning Approach: YOLOv5-based Custom Object Detection. Engineering, Technology & Applied Science Research, 13(6), 12158–12163.