International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Swiftcart: Revolutionizing Shopping Adventure Using Image Processing and Cloud Computing with IoT

B V Sai Kiran[1], S A Bhavani[2], B Likhit Sri Sai[3], A Nagendra[4], K Anupama[5], J Harshit[6], A Hemanth Kumar[7]

[1,3,4,5,6,7]Student, Computer Science and Engineering, Anil Neerukonda Institute of Technology & Sciences
[2]Assistant Professor, Computer Science and Engineering, Anil Neerukonda Institute of Technology & Sciences

Abstract
In the present retail landscape of India, long and tedious checkout lines have been a persistent challenge for shoppers in both malls and grocery stores. This project aims to redefine the shopping experience by leveraging the power of Image Processing and cutting-edge technologies. By seamlessly integrating Image Processing with IoT, Database Management, Cloud Computing, and Android Application, we propose a novel approach to revolutionize the shopping process. Our solution involves installing cameras in shopping carts, enabling customers to effortlessly add items to a virtual cart by simply showing them to the camera. Upon completion, a QR code is generated for swift payment, eliminating the need for traditional checkout queues. This project envisions a smoother, more efficient, and convenient shopping journey for customers in India.
The core of our system lies in the intelligent use of Image Processing technology. The cameras installed in the carts are equipped with advanced image recognition algorithms that instantly identify products and add them to a virtual cart. This data is seamlessly integrated into a robust Database Management System, where customers can review and manage their selections. Leveraging Cloud infrastructure ensures secure storage of organizational data and efficient scaling to accommodate growing user demands. Our Android Application offers a user-friendly interface for customers to view their virtual carts, modify selections, and generate a QR code for payment. With these integrated technologies, we envision transforming the traditional shopping paradigm, fostering greater convenience, reducing wait times, and enhancing overall customer satisfaction. Welcome to Swiftcart where the future of shopping begins.

Keywords: Image Processing, Internet of Things IOT, Robust Database Management System RDBMS, Convolutional Neural Network CNN, Cloud Computing, Android Application, QR code generation, Camera Installation, Virtual Cart, IN and OUT Algorithm.

1. Introduction

In the retail landscape of India, the perennial challenge of long and tedious checkout lines has persisted, often frustrating shoppers in both malls and grocery stores. The valuable time spent waiting has become

an unwelcome part of the shopping experience. Recognizing this pain point, we embarked on a visionary journey with Swiftcart, a transformative solution that seeks to redefine shopping through the integration of cutting-edge technologies. Swiftcart's core innovation revolves around the smart utilization of Image Processing technology, driven by powerful algorithms such as YOLOV5 and RCNN. These sophisticated image recognition algorithms are seamlessly embedded into our system. When customers place products into their shopping carts, the cameras adorning these carts instantly and accurately identify the items. Traditional manual scanning becomes a relic of the past, as customers effortlessly populate their virtual carts by merely presenting products to the camera. This streamlined process eradicates the frustration of laborious checkout queues.

Swiftcart takes a comprehensive approach by expertly integrating IoT devices, including Raspberry Pi, with the robust cloud computing capabilities of AWS. These IoT devices serve as the sensory eyes of the system, capturing real-time data from the shopping environment. The cloud infrastructure ensures secure data storage and offers scalable processing power. This symbiosis of the physical and digital realms amplifies the system's efficiency, allowing it to gracefully handle expanding user demands while upholding data security and integrity. Swiftcart envisions a shopping ecosystem where convenience reigns supreme, wait times are minimized, and overall customer satisfaction soars, marking the dawn of a new era in Indian retail. Swiftcart is more than a technological leap; it represents a holistic reimagining of the retail experience. We have seamlessly integrated state-of-the-art technologies, including image processing, IoT integration, cloud computing, and mobile application development, to craft a shopping ecosystem that transcends traditional boundaries. Our vision is to foster unparalleled convenience, reduce wait times, and elevate overall customer satisfaction. By eliminating the inefficiencies of checkout queues and empowering customers with the freedom to shop seamlessly, we strive to set new standards in the retail landscape of India. Swiftcart is where tradition harmonizes with technology, where the age-old frustrations of shopping find resolution, and where the future of retail takes root, promising a smoother, more efficient, and more satisfying shopping journey for all. Welcome to Swiftcart, where the retail experience you've always yearned for finally becomes a reality.

1.1 . Motivation and Problem statement

Swiftcart is propelled by the unwavering commitment to redefine the shopping landscape in India, specifically tackling the enduring challenge of lengthy checkout queues. Recognizing the inconvenience and dissatisfaction that traditional checkout processes bring to shoppers, our motivation stems from the desire to introduce groundbreaking solutions that leverage cutting-edge technologies. Among these, Image Processing takes center stage, with YOLO (You Only Look Once) serving as a pivotal tool for precise and rapid object detection. The amalgamation of these technologies seeks to elevate and streamline the overall shopping experience. The retail environment in India is damaged by the persistently tedious nature of checkout queues, representing a significant pain point for shoppers. The conventional checkout process, marked by extended wait times and operational inefficiencies, necessitates an innovative intervention. Swiftcart addresses this challenge comprehensively by strategically integrating cameras into shopping carts, offering a dynamic solution powered by advanced image recognition algorithms, prominently featuring YOLO. As customers navigate the aisles and add items to their carts, the system captures images in real-time, deploying YOLO to swiftly and accurately detect objects. The captured data seamlessly populates a virtual cart, which is intricately connected to cloud services, specifically AWS (Amazon Web

Services). This virtual cart serves as a digital representation of the items selected by the customer, allowing for continuous monitoring, modification, and management.

1.2. The main features of this research work are listed as follows

- **Addressing the Pressing Issue:** The project directly tackles the common and frustrating issue of long checkout lines in shopping malls and grocery stores, resonating with the daily experiences of a wide range of consumers.
- **Modernization of Shopping:** With the integration of Image Processing, IoT, and Cloud Computing, the project aligns with the ongoing trend of digital transformation, bringing a fresh perspective to traditional shopping practices.
- **Broad Applicability:** The concept has potential applications beyond retail, such as in other service sectors where minimizingwaiting times is crucial, making the project adaptable to various industries.
- **Positive Societal Impact:** Shorter wait times enhance customer satisfaction, positively affecting overall shopping experiences and contributing to a happier, less stressful society.

2. Related works

Hong-Chuan Chi, Muhammad Atif Sarwar, Yousef-Awwad Daraghmi, Kuan-Wen Liu, Ts`ı-U´ı ˙Ik, Yih-Lang Li proposed a prototype of a smart shopping cart based on image-based action recognition. Deep learning networks such as Faster R-CNN, YOLOv2, and YOLOv2-Tiny are utilized to analyze the content of each video frame. Frames are classified into three classes: Hands and No hand with only cart. The classification accuracy based on Faster RCNN, YOLOv2, or YOLOv2-Tiny is between 93.0% and 90.3%, and the processing speed of the three networks can be up to 5 fps, 39 fps, and 50 fps, respectively. Secondly, based on the sequence of frame classes, the timeline is divided into No Hand intervals, Empty Hand intervals, and Holding Items intervals. The accuracy of action recognition is 96%, and the time error is 0.119s on average. Finally, we categorize the events into four cases: No Change, placing, Removing, and Swapping.

Hong-Chuan Chi, Muhammad Atif Sarwar, Yousef-Awwad Daraghmi, Kuan-Wen Liu, Ts`ı-U´ı ˙Ik, Yih-Lang Li proposed a smart shopping cart with self-checkout, called iCart, to improve customer's experience at retail stores by enabling just walk out checkout and overcome the aforementioned problems. iCart is based on mobile cloud computing and deep learning cloud services. In iCart, a checkout event video is captured and sent to the cloud server for classification and segmentation where an item is identified and added to the shopping list. The Linux based cloud server contained the yolov2 deep learning network. iCart is a lightweight system of low-cost solutions which is suitable for small-scale retail stores. The system is evaluated using real world checkout video, and the accuracy of the shopping event detection and item recognition is about 97%.

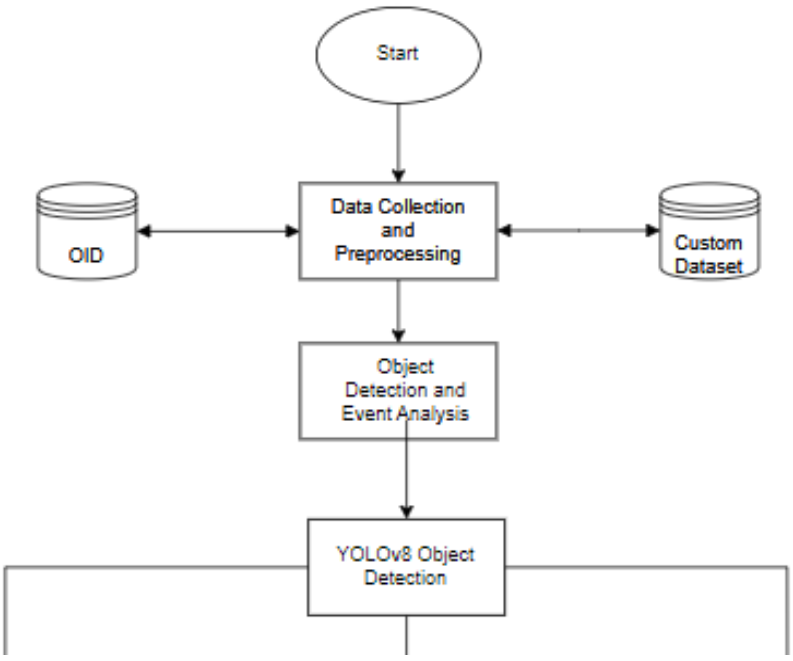
Roopa C, Nivas Chandra Reddy took an example of DMart which is a well-known retail enterprise in India, seeing at the statistics of DMart billing time analysis form 4:00 PM to 8:00 PM the percentage of the customers at the billing queue is increased from 13.73% to 47.06%. Considering this they proposed a project which presents a plan to build up a framework in shopping centers to beat the above issue. To realize this all merchandise within the mall should be equipped with RFID tags and every one trolley should be equipped with an RFID reader and digital display screen. When one puts any item in the trolley its code will be distinguished naturally, the item name and cost will be shown on the LCD, in this manner the expense gets added to the absolute bill. On the off chance that we wish to expel the item from the

trolley, you can remove the item and the measure of that particular item gets deducted from the aggregate sum and a similar data goes to the central billing unit through ZigBee module. Subsequently the billing should be possible in the trolley itself.

Researchers Rajesh Kannan Megalingam, Souraj Vishnu, Swathi Sekhar, Vishnu Sasikumar, Sreekumar S, and Thejus R Nair noted that individuals use technology to solve difficulties in their daily lives and live more innovative lives. When it comes to shopping, consumers struggle to find what they're looking for in the store and to wait in lengthy lines at the billing counter. Here, they develop an Android application that addresses these issues and offers a better shopping experience. This program can be utilized in smart operating shopping carts. Their two-part smart software is primarily designed for navigating to the item's location and automatically paying the user for the things they have purchased. The Android Studio software is an open-source program that serves as the app's development environment. The products are scanned using an RFID reader. The design, implementation, and outcomes of the application are covered in this document.

3. Proposed methodology

The proposed solution envisions a revolutionary shopping experience seamlessly blending cutting-edge technologies. At its core, advanced image processing, powered by Convolutional Neural Networks (CNNs), empowers smart shopping carts with real-time product recognition. These IoT-enabled carts, equipped with high-resolution cameras and microcontrollers, communicate effortlessly with a cloud infrastructure hosted on AWS. Leveraging Amazon S3 for image storage, AWS Lambda for serverless microservices, and RDS for efficient database management, the system ensures scalable and secure data processing. The user-friendly mobile application, crafted for both iOS and Android platforms, facilitates dynamic interactions, allowing customers to effortlessly manage their virtual carts. Security measures, including HTTPS, secure coding practices, and regular updates, safeguard user data. Figure 3.1 manipulates the workflow of the proposed method.



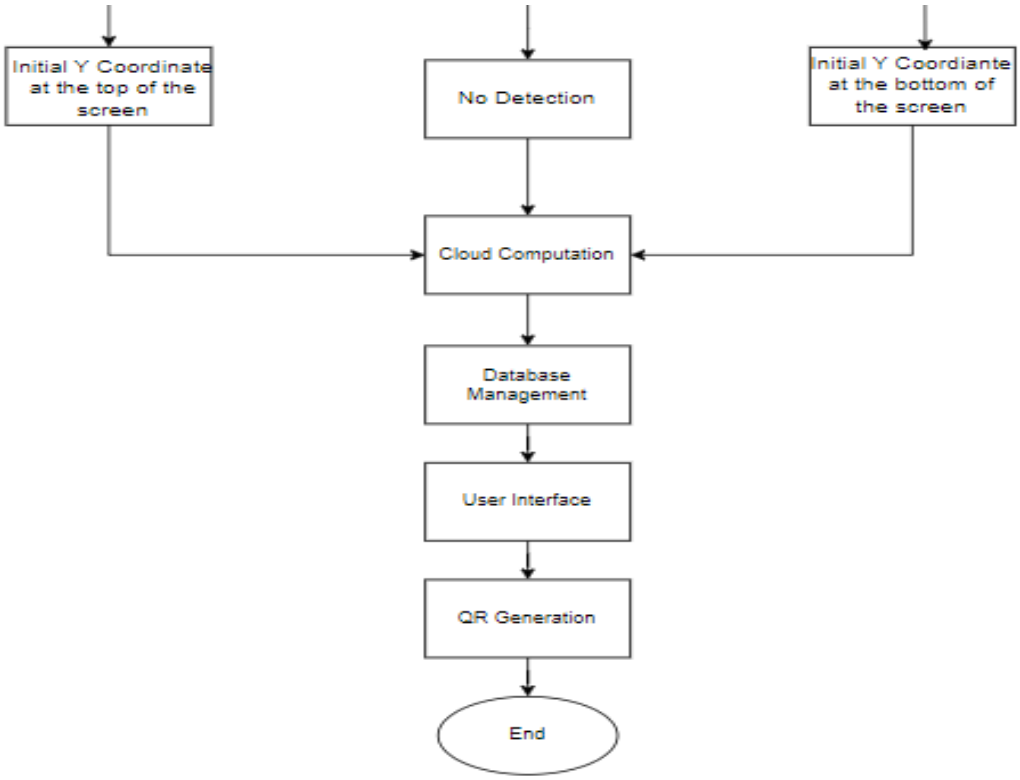


Fig- 3.1 System Architecture of the proposed model

3.1. Modules

3.1.1. Data Collection and Preprocessing

Data Collection and Preprocessing in the Swiftcart project involve the systematic acquisition and initial preparation of image data from cameras mounted on shopping carts. This critical step ensures that the data obtained from D-Link or Logitech webcams, integrated with Raspberry Pi devices, is of high quality and suitable for subsequent image recognition processes. Data collection mechanisms capture real-time images of products as they are placed in the shopping carts, and preprocessing tasks involve tasks such as image resizing, noise reduction, and format standardization to optimize the data for efficient processing by the Image Processing module.

In this phase of the project, a crucial preprocessing pipeline is implemented to optimize the input data for advanced image processing algorithms. The process begins with the acquisition of high-quality images using cameras embedded in shopping carts, ensuring proper calibration and lighting conditions. Subsequently, a series of steps are executed to enhance the clarity and uniformity of the dataset. Noise and artifacts are meticulously removed through the application of filters, and images are resized to a standardized resolution. The cropping process focuses on isolating the relevant regions containing the products of interest, optimizing the dataset for accurate recognition. Adjustments to contrast and brightness further improve visibility, and normalization techniques ensure consistent input values for subsequent image recognition algorithms. While the presence of a reference image or information regarding the reference simplifies the problem of quality assessment, practical applications of such algorithms are limited in real-world scenarios where reference information is generally unavailable at nodes where quality computation is undertaken [1]. Additionally, data augmentation introduces diversity into the dataset by applying random transformations, enhancing the model's robustness. The dataset is

strategically split into training and validation sets, facilitating comprehensive model evaluation and ensuring balanced class representation. Through these preprocessing steps, the project establishes a foundation of clean and standardized data essential for the success of the intelligent image processing system

3.1.1.1. Image Denoising

Numerous image smoothing approaches, such as Gaussian blurring and median blurring, have been demonstrated, and they have been somewhat successful in reducing minor amounts of noise. In those methods, the central element was substituted by taking a tiny neighbourhood surrounding a pixel and performing various operations on it, such as gaussian weighted average, median of the values, etc. To put it briefly, noise reduction at a pixel was limited to its surrounding area. Using convolutional neural networks (CNN) can significantly improve the function of image denoising techniques [2].

Noise has a certain quality. Most people believe that noise is a random variable with zero mean. Think of a noisy pixel as p = p0 + n, where n represents the noise in the pixel and p0 is its true value. You may calculate the average of a huge number of identical pixels (let's say N) from various photos. Ideally, since the mean of the noise is zero, you should obtain p = p0. Denoising is an essential and widely used technique for image processing that has been extensively studied and numerous related techniques, such as the wavelet, Wiener, and Gaussian filters, have been presented. [3][4][5][6][7].

3.1.1.2. Spatial domain filtering

Since filtering is a common method of image processing, image denoising has made use of a wide range of spatial filters, which are further divided into two categories: linear and non-linear filters.

Although linear filters were first used to eliminate noise in the spatial domain, they are unable to maintain the textures of images. Although mean filtering has been used for Gaussian noise reduction, photographs with high noise may be oversmoothed [1]. Wiener filtering has been used more often to get around this drawback, but it can also readily blur sharp edges. Noise can be suppressed without identification by employing non-linear filters, such as weighted median filtering and median filtering. Bilateral filtering is a non-linear, edge-preserving, noise-reducing smoothing filter that is frequently used for image denoising. A weighted average of the intensity values from neighbouring pixels is used to replace each pixel's intensity value [2]. The efficacy of the bilateral filter is one problem. O(n^2) time is required for the brute-force implementation, which is prohibitively high when the kernel radius *r* is large [1].

Low pass filtering is applied to pixel groups using spatial filters, which claim that the noise is located in a higher frequency spectrum. Spatial filters typically reduce noise to an acceptable level, but at the expense of blurring the image and losing its crisp edges [5].

3.1.2. Object Detect and Event analysis

The Object Detection and Event Analysis component in the Swiftcart system play a pivotal role in enhancing the shopping experience. This module employs advanced image processing algorithms, such as YOLOV5, to accurately detect and identify products in real-time as they are placed in shopping carts. Beyond product identification, it also performs event analysis, which involves recognizing significant events like holding items, empty hands, and instances where no hands are detected.

Technologies Used

3.1.2.1 YOLOV5

YOLOV5, short for "You Only Look Once version 5," is an advanced real-time object detection and recognition algorithm. It belongs to the YOLO family of models, which are renowned for their speed and accuracy in identifying objects within images or video streams. YOLOV5 is characterized by its ability to detect multiple objects in a single pass, making it highly efficient and suitable for various applications, including autonomous vehicles, surveillance, and robotics [9]. It employs deep learning techniques, typically based on convolutional neural networks (CNNs), to analyze and classify objects within a scene. YOLOV5's continuous development and refinement have made it a popular choice in computer vision tasks, where rapid and accurate object detection is critical. Popular algorithms of object detection include You Only Look Once (YOLO), Region-based Convolutional Neural Networks (RCNN), Faster RCNN (F-RCNN). RCNN has better accuracy compared to other algorithms but YOLO surpasses when speed is considered over accuracy [11]. YOLO suffers from a variety of shortcomings relative to state-of-the-art detection systems. Error analysis of YOLO compared to Fast R-CNN shows that YOLO makes a significant number of localization errors [10].

Working Principle of YOLOV5

YOLOV5 works by first dividing the input image into a grid of cells. For each cell, YOLOV5 predicts a set of bounding boxes, along with the class probabilities for each bounding box YOLOV5 then uses a non-maxima suppression (NMS) algorithm to filter out overlapping bounding boxes and select the most likely bounding boxes for each object in the image. COCO (Common Objects in Context) is the industry standard benchmark for evaluating object detection models. Furthermore, YOLO has relatively low recall compared to region proposal-based methods [10]. When comparing models on COCO, we look at the mAP value and FPS measurement for inference speed. Models should be compared at similar inference speeds.YOLOV5 COCO accuracy is state of the art for models at comparable inference latencies as of writing this post. Difference between variants of Yolo V8: YOLOV5 is available in three variants: YOLOV5, YOLOV5-L, and YOLOV5-X. The main difference between the variants is the size of the backbone network. Responding to the need for real-time applications, the YOLO, YOLOv2, and YOLOv3 are an end-to-end design that achieves high processing speed. YOLO splits the input image into S × S grid cells and extracts feature using the CNN [9]. YOLOV5 has the smallest backbone network, while YOLOV5-X has the largest backbone network. The larger backbone network in YOLOV5-X gives it better accuracy, but it also makes it slower than YOLOV5 and YOLOV5-L.

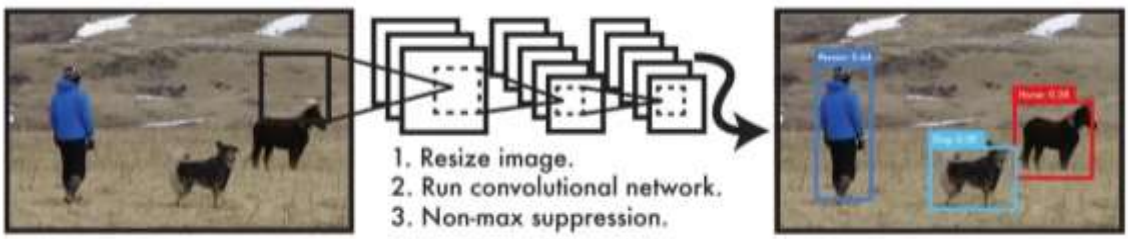


Fig – 3.2 Working of Convolutional Neural Network

Implementing YOLOV5

Let us look at how to use and implement YOLOV5 into your workflows. The model comes bundled with

the following pre-trained models that can be utilized off-the-shelf in your computer vision projects to achieve better model performance:

- Instance segmentation models trained on the COCO segmentation dataset with an image resolution of 640.
- Image classification models pre-trained on the ImageNet dataset with an image resolution of 224.
- Object Detection models trained on the COCO detection dataset with an image resolution of 640.

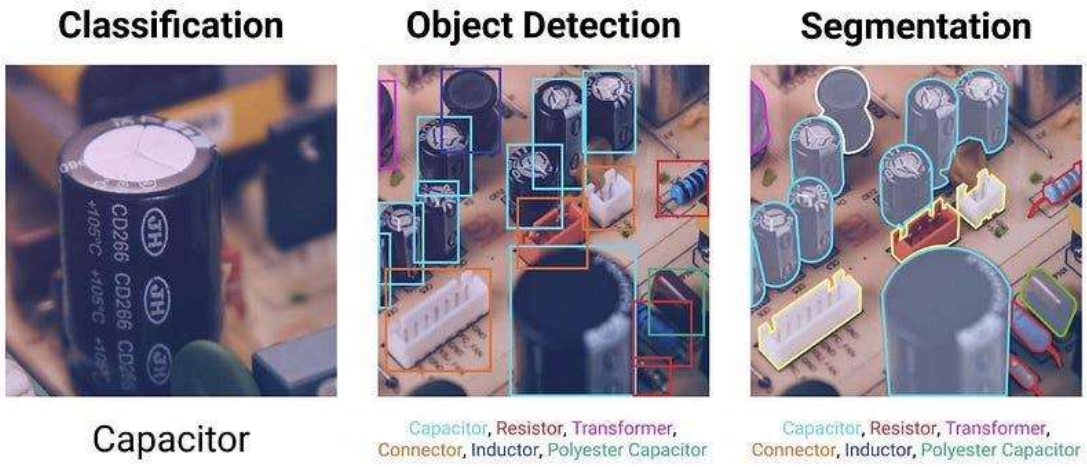


Fig – 3.3 Categories of Image Processing (a) Classification (b) Object Detection (c) Segmentation

3.1.2.2 Pyzbar

The barcode detection process in your system begins with capturing frames from the webcam using OpenCV. These frames are then fed into the YOLOv5 object detection model, which identifies various objects, including barcodes, within the frames. Once objects are detected, the system extracts the regions containing the barcodes and converts them to grayscale for efficient barcode decoding. The pyzbar library is employed to decode the barcodes from the grayscale images. This library scans the images for barcodes and returns the decoded data, which typically consists of product IDs or other encoded information. The decoded data is then utilized for tasks such as looking up item information in a database or performing specific actions based on the scanned items. Overall, this process enables automated barcode scanning and interaction, facilitating tasks such as inventory management or product checkout in our Swiftcart project [12].

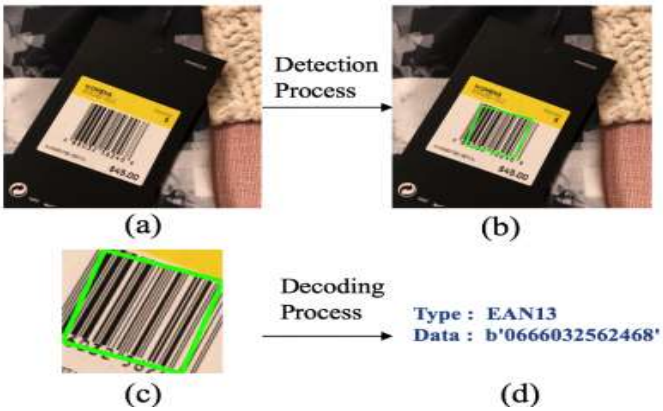


Fig 3.4 Decoding Process of Barcode

3.1.2.3. RCNN

RCNN (Region-based Convolutional Neural Network) can be employed in the Swiftcart project to enhance the accuracy of product recognition and event analysis. By leveraging RCNN, the system can efficiently localize and classify objects within the images captured by shopping cart cameras. This technology can complement YOLOV5 by providing finer-grained object recognition, ensuring that even small or complex items are identified accurately. Moreover, RCNN can aid in event analysis by detecting and tracking hands, empty hands, and no hands in real-time, contributing to a more responsive and interactive shopping experience. Integrating RCNN into the existing image processing pipeline can refine the system's ability to recognize and analyze objects and events, ultimately elevating the convenience and efficiency of Swiftcart for customers in India. The proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection [13]. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher map on PASCAL VOC 2012 Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate [13].

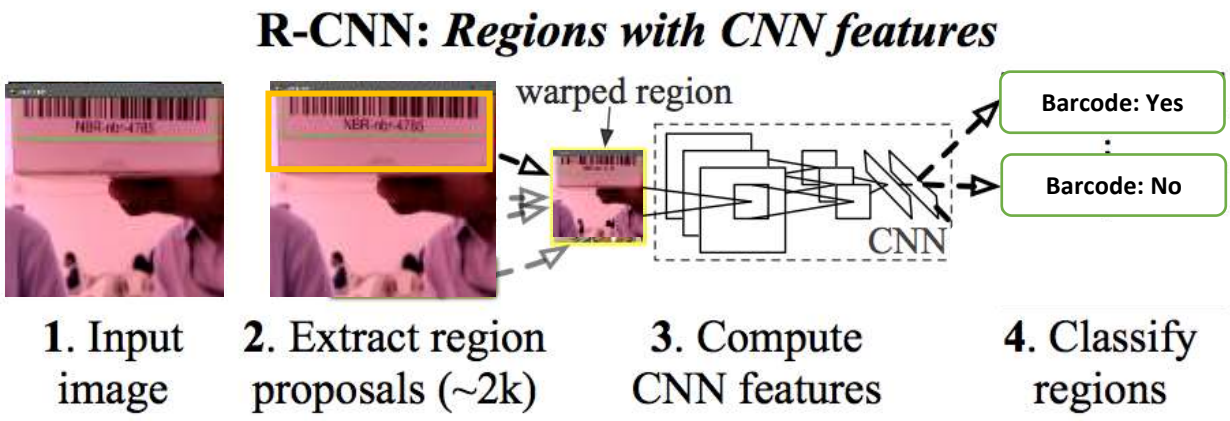


Fig – 3.5 Process flow of the input by RCNN

3.1.2.4. Convolutional Neural Network (CNN)

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Convolutional Neural Network has had ground breaking results over the past decade in a variety of fields related to pattern recognition from image processing to voice recognition [8]. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications take an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension) [8]. E.g., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1 [8]. The below figure is a complete flow of

CNN to process an input image and classifies the objects based on values.[1]

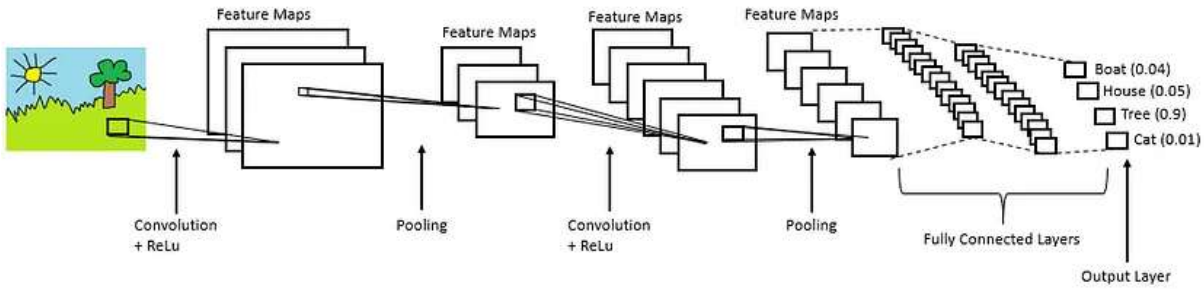


Fig – 3.6 Architecture of Convolutional Neural Network

3.1.3. Cloud Computing and Image Data transfer using IOT/Webcam

Cloud computation in the Swiftcart system is a fundamental component that enhances the efficiency and capabilities of the platform. While Raspberry Pi/ Webcam devices are employed for capturing image barcodes and videos from cameras mounted on shopping carts, the true power of image processing and recognition is unlocked in the cloud. Once the data is sent to the cloud, it undergoes intensive processing using state-of-the-art technologies like YOLO (You Only Look Once), Pyzbar, and RCNN (Region-based Convolutional Neural Network). These cloud-based algorithms facilitate real-time product recognition, hand gesture analysis, and event detection.

3.1.3.1 Raspberry Pi (Extension)

Raspberry Pi serves as a vital component in the Swiftcart project by facilitating real-time data collection from cameras mounted on shopping carts. These compact, affordable devices are integrated with the cameras to capture images or video feeds, which are then processed and transmitted to the cloud for advanced image recognition and analysis. The Raspberry Pi is a credit card sized computer developed by the Raspberry Pi Foundation. The RPi's ability to act as a GNU/Linux server and the interfacing services provided by its general purpose I/O pins make it a popular choice of hardware for IoT applications [15]. Raspberry Pi plays a pivotal role in bridging the physical and digital shopping realms, enabling the system to identify products, track user actions, and offer a seamless, intelligent shopping experience that eliminates traditional checkout queues, benefiting customers and retailers alike in India. A sensor will be attached to the RPi; the RPi will be responsible for taking the data from the sensor and then using the CoAP protocol to transmit this data to the cloud platform [15].

3.1.4 User Interface using Android application

The User Interface (UI) implemented through the Android application is a central element of the Swiftcart system, providing a seamless and user-friendly shopping experience. The captured and detected images, combined with advanced image processing algorithms, enable the app to recognize user actions such as adding, removing, or swapping items in their shopping cart. As users interact with the physical shopping environment, the app dynamically updates the virtual cart displayed on their mobile device.

3.1.4.1 Android (JAVA)

The Android application in our Swiftcart project acts as the primary interface for customers, enabling them to effortlessly manage their virtual shopping carts. It allows users to add, remove, or swap items based on

real-time product recognition and gestures captured by shopping cart cameras. Virtual Cart is a lightweight system of low-cost solution which is suitable for the small-scale retail stores. The system is evaluated using real world checkout video, and the accuracy of the shopping event detection and item recognition is about 97% [14]. The app also generates QR codes for swift payments, streamlining the checkout process. With its intuitive and responsive user interface, the Android application plays a central role in revolutionizing the shopping experience, making it convenient, efficient, and enjoyable for users in India.

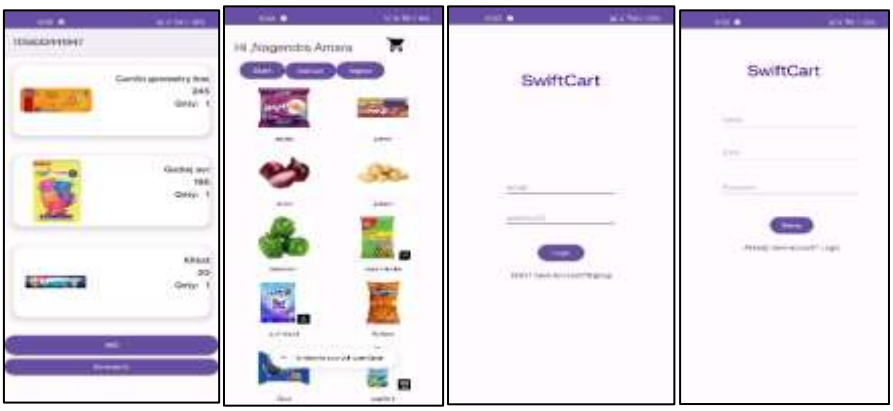


Fig – 3.7 Interfaces of Swiftcart Application

3.1.5 QR Generation

Towards the culmination of the Swiftcart shopping experience, a QR code is generated within the Android application to facilitate a swift and efficient checkout process. Nowadays, mobile phones with built-in camera are wildly used to recognize the QR code [16]. Once the user has completed their shopping and is ready to make the payment, they simply present the generated QR code to the cash counter person. When the cash counter person scans the QR code, all the details of the items in the virtual cart, along with the corresponding bill, are instantly displayed on the counter's screen. This seamless transition from the virtual cart to the payment process ensures that the user experiences a hassle-free checkout, with all relevant information readily available for review and payment. It streamlines the traditionally time-consuming checkout queues, providing a convenient and efficient payment process for both customers and retailers in India.

3.1.6 Virtual Shopping Cart

Finally, the virtual shopping cart module calculates the item list according to the detected shopping events. The item list is displayed on the smartphone app via the Google FCM service for reference. The virtual shopping cart module displayed the items as it received the tokens from the server. And at last a security scrutiny will be present to check whether all the items were added to the virtual cart and then payment process will be done directly without any scanning process.

3.1.7 Testing and Evaluation

Swiftcart undergoes rigorous testing at various levels, including unit testing, integration testing, and load testing. Unit tests validate the functionality of individual components, such as image recognition algorithms and database interactions. Integration tests verify the seamless interaction between different modules, such as IoT devices, cloud services, and the Android application. Load testing assesses the

system's ability to handle increasing user loads and ensures that it remains responsive under various conditions. Real-world scenarios are simulated to identify and address potential bottlenecks or performance issues.

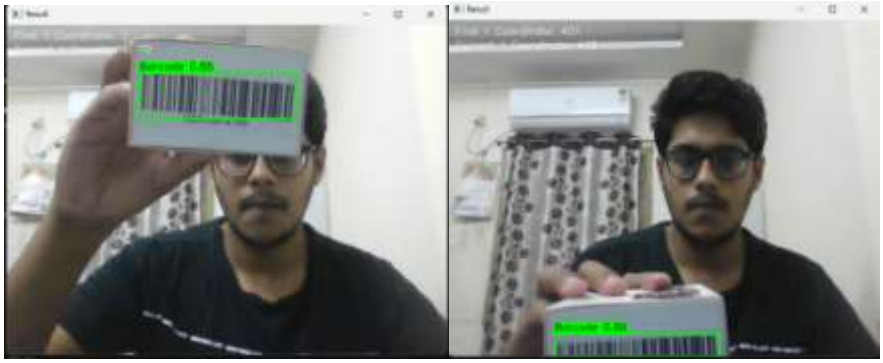
3.2 Algorithm for event analysis

3.2.1 Cart Video Module

The cart video module monitors and records shopping activities of the cart. A Customer starts a video of shopping activities, particularly items added or removed from the cart, once enters the store using the app installed on the smartphone. The smartphone supports the cart video module. After finishing the shopping, the Customer transfers the videos to back-end servers for further processing. Here video as in camera captures frames and send each frame for evaluation.

3.2.2 Frame Classification Module

This system classifies frames based on the movement of items within the images captured by the webcam, utilizing deep learning algorithms. The classification process hinges on discerning the direction of item movement concerning a reference point, typically the initial y-coordinate compared to the current y-coordinate. Frames are categorized into different classes depending on this comparison: frames where items are descending relative to the initial y-coordinate are classified as items being added (ADD), while frames where items are ascending are classified as items being removed (REMOVE). Frames where no significant item movement is detected relative to the initial y-coordinate are classified as no action (NO ACTION). This approach enables automated tracking and management of items passing through the webcam's field of view, facilitating tasks such as inventory updates or checkout processes in the Swiftcart project.



(a) **(b)**

Fig – 3.8 Categories of Activities (a) First Y coordinate is near to top of the screen (First Y Coordinate: (0))
(b) First Y coordinate is near to bottom of the screen (First Y Coordinate: (451)

3.2.3 Shopping Action Recognition Module

The shopping action analysis in this system relies on the detection of item movements using the initial and final y-coordinates. When an item is detected, its barcode is recognized, and the initial y-coordinate is stored. Subsequently, when the final y-coordinate increases compared to the initial y-coordinate, it indicates that the item is being added to the cart, whereas if it decreases, the item is considered to be removed from the cart. This process partitions the timeline into action intervals based on consecutive frames with consistent item movement directions.

To mitigate misclassifications of frames, a smoothing algorithm is employed, which considers a window of k frames before and after each frame to determine its class, effectively correcting misclassifications and removing fragmented intervals caused by such errors. In the final evaluation, a value of k = 2 is selected for optimal performance. The major vote based smoothing algorithm is described below.

3.2.4 Smoothing Algorithm for Frame Classification

3.2.4.1 Smooth_frames Function

Smoothing algorithm is used to rectify and filter the frames which are noise and which are already computed. To make our cloud capture and run large machine learning and deep learning model we must make sure we are handling all the produced and pre-installed noise and also transferring the images which are of most need. The below pseudocodes are referring to the algorithm of the smoothing concept.

Input: frames - a list of frames, k - a parameter indicating the window size.
Output: smoothed_frames - a list of frames after smoothing.
a. Initialize an empty list smoothed_frames.
b. Iterate through each frame in frames.
c. For each frame:
(i) Retrieve the current frame.
(ii) Check its consistency using the check_frame_consistency function.
(iii) If the frame is considered valid, append it to the smoothed_frames list.
d. Return the list of smoothed_frames.

(a) (k-2) Frames (b) Current frame (c) (k+2) Frames

Fig – 3.9 Representation of Smoothing Algorithm

3.2.4.2 Check_frame_consistency Function

Input: frames - the list of frames, index - the current frame index, k - the window size.
Output: True if the frame is consistent, False otherwise.
a. Retrieve the current frame at the given index.
b. Iterate over a range of frames within the window [max(0, index - k), min(length(frames), index + k)].
c. For each frame in the window:
(i) Check if the current frame matches the frame within the window using the frames_match function.
(ii) If any frame does not match, return False.
d. If all frames match, return True indicating that the current frame is consistent.

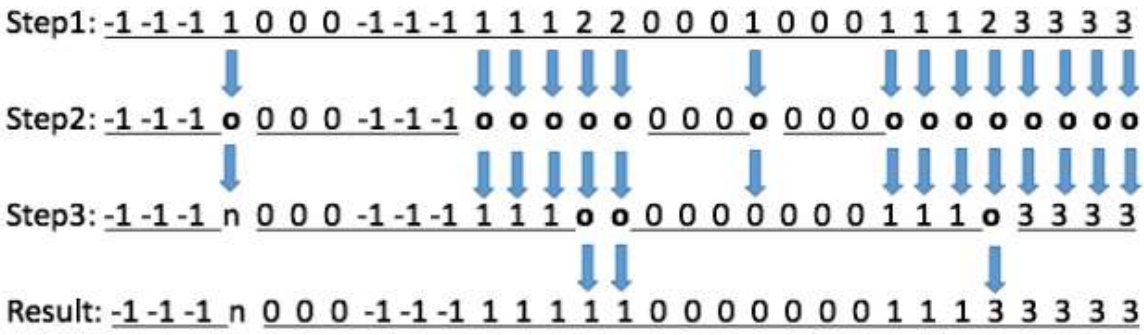


Fig – 3.10 Removing the unwanted frames by locating noise

3.2.4.3 Frames_match Function

Input: frame1 - the current frame, frame2 - a frame within the window.
Output: True if the frames match, False otherwise.
a. Implement the logic to compare whether two frames match.
b. This can include comparing pixel values, features, or any other relevant information based on the nature of your frames.
c. Return True if the frames match, False otherwise.

3.2.5 Shopping Event Detection Module

In this system, the shopping event is determined by changes in item presence between the detection and absence of barcodes, utilizing the initial and final y-coordinates of items. These events encompass no change, placing, and removing of items. "No change" indicates that the items within the cart remain unaltered. Possible scenarios include (1) items being added and removed when the barcode is initially detected and disappears respectively, and (2) items being added and removed when the barcode is detected and disappears while the item is being held. "Placing" signifies adding an item into the cart when the barcode is initially detected, and "Removing" denotes removing an item from the cart when the barcode disappears. By monitoring the items involved in these actions, the system maintains an updated list of items in the cart, adjusting quantities accordingly. For example, if an event involves placing item 'A' into the cart, the quantity of item 'A' in the shopping list will increase by one.

4. Results and discussion

4.1 Dataset of target images

The crucial portion of this process is to gather the target images for training purpose. During the detection the model will perform an algorithm which resembles to the one which we trained before. Many factors will affect the image processing with the training images and actual real time captured images. We need to look into lightings, shadows and many factors. The methodological level of the related work includes the methods used for objects and event recognition. Traditional image recognition often utilizes the histogram of oriented gradients (HOG) [9]. HOG plays important role to classify the images with different illumination The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. We study the issue of feature sets for human detection, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets including wavelets [17].

Fig – 4.1 Dataset of target images

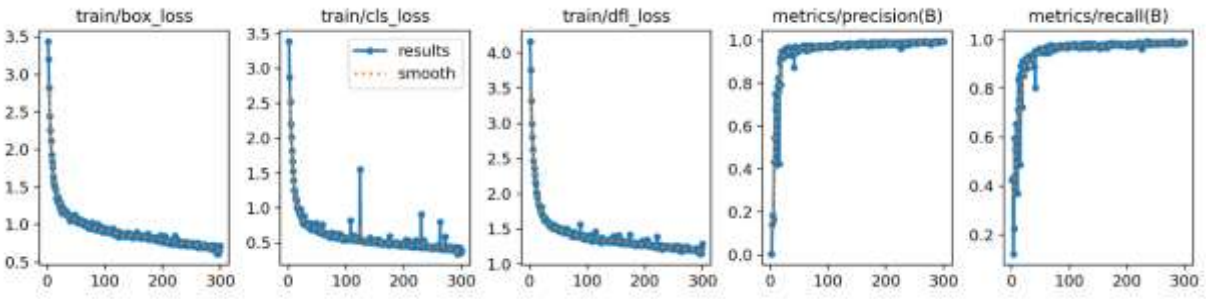
These are the images of Barcodes captured of nearly 800 images under different illumination conditions. Similarly capture various grocery items.

4.2 YOLOV5n

The YOLOV5n here n refers to nano, the code utilizes the Ultralytics library to train a YOLOV5 object detection model on a custom dataset of images containing a single class ('oreo1','nutella1','ariel'…). The dataset is organized into training, validation, and test sets located in specified directories. The YOLO model is configured using a YAML file ('YOLOV5n.yaml'). The training process is set to run for 300 epochs, aiming to optimize the model's performance on the provided data. The dataset information, such as the number of classes ('nc') and class names ('['oreo1']'), is specified. Additionally, the code integrates with the Roboflow platform for dataset management, referencing a specific workspace, project, and version. The provided URL points to the Roboflow dataset for further exploration. In summary, the code demonstrates the training of a YOLOV5 model on a custom dataset, leveraging the Ultralytics library and integrating with Roboflow for dataset versioning and management.

4.3 Evaluation and Results

Besides the bounding box coordinates of a detected object, the output also includes the confidence level and its class. lot for different detection algorithms [18]. Bounding-box detections are mostly represented by their top-left and bottom-right coordinates (xini, yini, xend, yend), with a notable exception being the



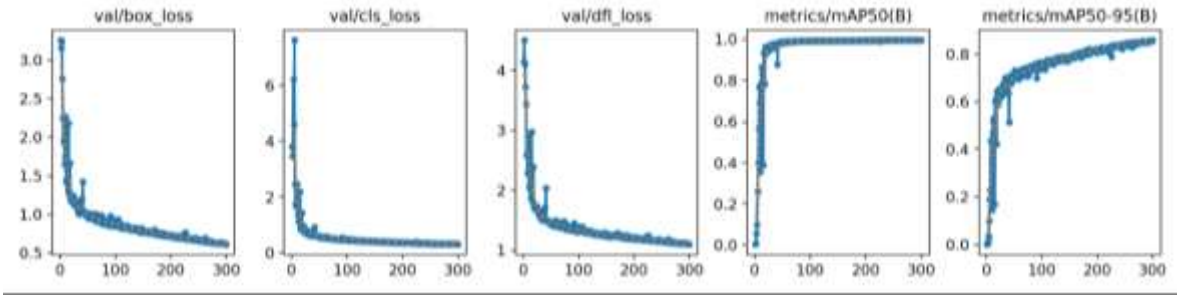


Fig – 4.2 Evaluation and Results of loss and mAP functions

YOLO algorithm that differs from the others by outlining the bounding boxes by their center coordinates, width, and height $(\frac{x\_center}{image\_width}, \frac{y\_center}{image\_width}, \frac{x\_max}{image\_width}, \frac{x\_min}{image\_width})$.

Here are the results from the training model which shows the Loss functions and MAP (Mean Average Precision). The loss functions as you can see in the first six blocks of the graphs, as a standard satisfactory term the curve should be leaned downwards as it depicted in the above graphs. To make it clear thick of it like the "LOSS" need to be decrease chronologically.

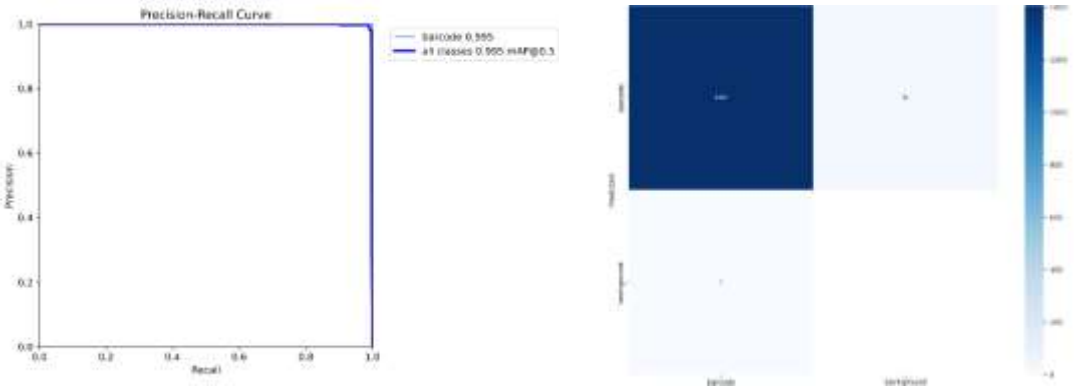
4.3.1 mAP (Mean Average Precision)

The mean AP (mAP) is a metric used to measure the accuracy of object detectors over all classes in a specific database. The mAP is simply the average AP over all classes, i.e.

$$\text{mAP} = \frac{1}{N}\sum_{k=1}^{n} \text{APk}$$

4.3.2 Precision Recall curve

Precision is the ability of a model to identify only relevant objects. It is the percentage of correct positive predictions. Recall is the ability of a model to find all relevant cases (all ground-truth bounding boxes). It is the percentage of correct positive predictions among all given ground truths. The precision × recall curve can be seen as a trade-off between precision and recall for different confidence values associated to the bounding boxes generated by a detector. If the confidence of a detector is such that its FP is low, the precision will be high. However, in this case, many positives may be missed, yielding a high FN, and thus a low recall. Conversely, if one accepts more positives, the recall will increase, but the FP may also increase, decreasing the precision. However, a good object detector should find all ground-truth objects (F N = 0 ≡ high recall) while identifying only relevant objects (F P = 0 ≡ high precision).



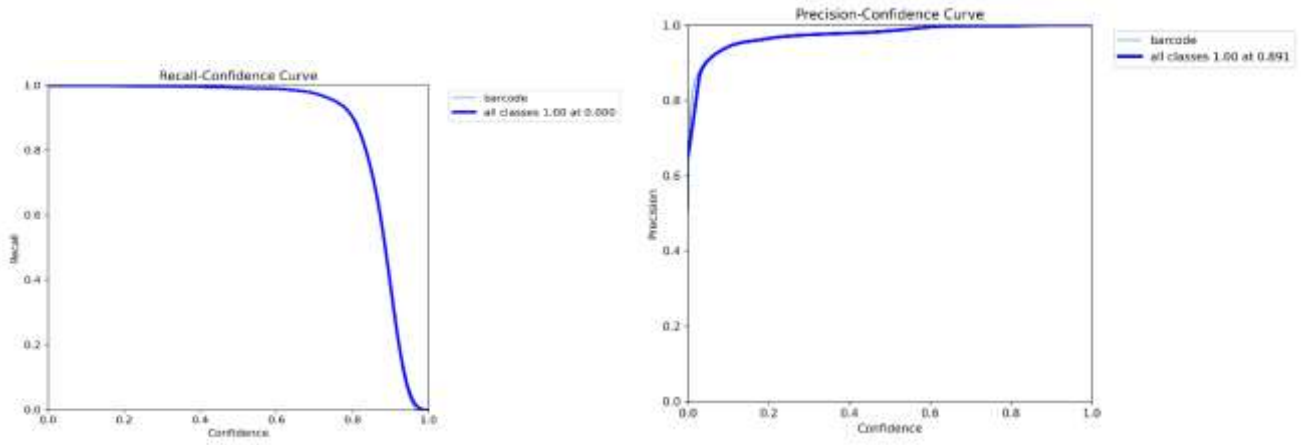


Fig – 4.3 Results of PR curve and Confusion matrix

4.4 Input and Expected output

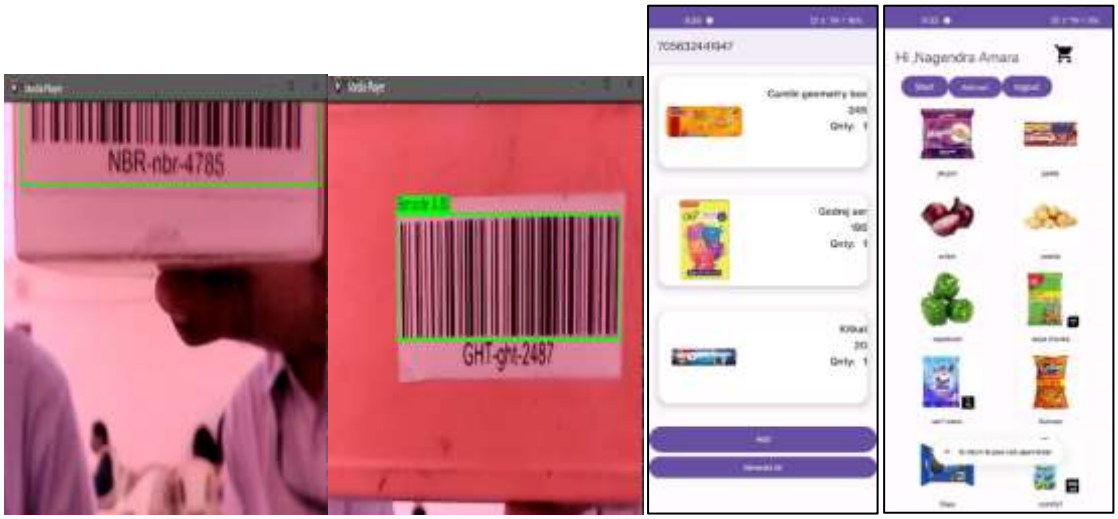


Fig – 4.4 Images captured through webcam and the payment gateway interface of the application

5. Conclusion

In conclusion, our project stands at the forefront of transforming the traditional shopping experience in India. By ingeniously amalgamating Image Processing, IoT, Cloud Computing, and Mobile App Development, we envision a future where the hassle of long checkout lines becomes a thing of the past. The sophisticated Image Processing algorithms, seamlessly integrated into smart shopping carts, bring forth a new era of efficiency by recognizing products in real time. The synergy with IoT devices facilitates instantaneous data transfer, allowing customers to effortlessly build virtual shopping carts. Our robust Cloud Computing infrastructure ensures secure and scalable storage, accommodating the evolving needs of a growing user base. The intuitive mobile application, designed for both iOS and Android platforms, becomes the gateway for customers to manage their selections, providing a streamlined and convenient shopping journey. As we usher in Swiftcart, the dawn of a novel shopping paradigm unfolds, promising not only greater convenience and efficiency but a transformative experience that resonates with the evolving demands of the modern Indian shopper. Welcome to the future of shopping, where innovation meets the everyday, and every checkout is a swift, satisfying experience.

References

1. No-Reference Image Quality Assessment in the Spatial Domain Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik, Fellow, IEEE
2. A Review and Taxonomy of Image Denoising Techniques Rusul Sabah Jebur, Chen Soong Der, Dalal Abdulmohsin Hammood
3. M. Malik, F. Ahsan, and S. Mohsin, "Adaptive image denoising using cuckoo algorithm," Soft Comput., vol. 20, no. 3, pp. 925–938, 2016.
4. N. D. Hoang, "An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with Steerable Filter-Based Feature Extraction," Adv. Civ. Eng., vol. 2018, 2018.
5. Y. Xian et al., "Data-driven tight frame for cryo-em image denoising and conformational classification," 2018 IEEE Glob. Conf. Signal Inf. Process., pp. 544–548, 2018.
6. L. Pfister and Y. Bresler, "Automatic parameter tuning for image denoising with learned sparifying transforms," IEEE Int. Conf. Acoust. Speech, Signal Process. 2017, no. March, pp. 6040–6044, 2017.
7. X. Luo and T. Bhakta, "Estimating observation error covariance matrix of seismic data from a perspective of image denoising," Comput. Geosci., vol. 21, no. 2, pp. 205–222, 2017.
8. Understanding of a Convolutional Neural NetworkAlbawi, Saad; Mohammed, Tareq Abed; Zawi, Saad (2017). [IEEE 2017 International Conference on Engineering and Technology (ICET) - Antalya, Turkey (2017.8.21-2017.8.23)] 2017
9. Hong-Chuan Chi "Smart Self-Checkout Carts Based on Deep Learningfor Shopping Activity Recognition "Muhammad Atif Sarwar Yousef-Awwad Daraghmi Kuan-Wen Liu† Ts`ı-U´ı ˙Ik Yih-Lang Li.
10. Joseph Redmon, Ali Farhadi "YOLO9000: Better, Faster, Stronger"University of Washington, Allen Institute for AI†, XNOR.ai
11. N. Murali Krishna "Object Detection and Tracking Using Yolo" Ramidi Yashwanth Reddy, Mallu Sai Chandra Reddy Department of CSE, Vignan Institute Of Technology & Science
12. Pyzbar: Barcode Detection and Decoding in On-line Fashion Images.
13. Fast R-CNN Ross Girshick, Microsoft Research
14. Smart Shopping Carts Based on Mobile Computing and Deep Learning Cloud Services Muhammad Atif Sarwar, Yousef-Awwad Daraghmi, Kuan-Wen Liu, Hong-Chuan Chi, Ts`ı-U´ı ˙Ik, Yih-Lang Li.
15. Thomas Lee Scott "CoAP based IoT data transfer from a Raspberry Pito Cloud" Amna Eleyan School of Computing, Mathematics & Digital Technology Manchester Metropolitan University Manchester, United Kingdom
16. Phaisarn Sutheebanjard "QR-code generator" Wichian Premchaiswadi
17. Navneet Dalal and Bill Triggs "Histograms of Oriented Gradients for Human Detection" INRIA INRIA Rhône-Alpes, 655 avenue de l 'Europe, Montbonnot 38334, France
18. Rafael Padilla "A Survey on Performance Metrics for Object-Detection Algorithms", Sergio L. Netto Eduardo A. B. da Silva 3PEE, COPPE, Federal University of Rio de Janeiro