SMART SHOPPING CART

A PROJECT REPORT

DE-39 (DC&SE)

Submitted by

NS MUHAMMAD OMER

NS MUHAMMAD HAMZA REHMAN

NS ZAIN NAQVI

PC SYED AHMED AKMAL

BACHELORS

IN

COMPUTER ENGINEERING

YEAR

2021

PROJECT SUPERVISOR

DR. USMAN AKRAM

COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING

PESHAWAR ROAD, RAWALPINDI

DECLARATION

We hereby declare that no portion of the work referred to in this Project Thesis has been submitted in support of an application for any other degree or qualification of this for any other university. If any act of plagiarism found, we are fully responsible for every disciplinary action taken against us depending upon the seriousness of the proven offence.

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

First and foremost, Alhamdulillah, our FYP has finally been completed, and we are grateful to Allah for providing us with the strength and morale to continue pushing on and for assisting us at every step of the road.

Second, we would like to express our heartfelt gratitude to our supervisor, Dr. Usman Akram, who helped us a lot, greatly, on every single subject, and whose assistance and direction became a source of great resolve for us. Thank you, sirs; you played an important part in our lives that we will never forget.

Finally, we would like to thank our parents and friends for their amazing support and persistent encouragement, without which we would not have been able to accomplish our Final Year Project. They performed an unrivaled role throughout our trip, for which we will be eternally grateful. Their unwavering support pushed us to do more than we could have imagined, and they instilled fresh hope in us when we had lost all hope in ourselves.

ABSTRACT

Numerous endeavours have as of late been made to diminish the measure of time it takes to pay in various retail settings. Besides, as the Fourth Industrial Revolution advances, man-made reasoning examination advances, and IoT PCs become more convenient and more affordable. Because of joining these two developments, it got conceivable to establish an automated climate for people to save clients' time. We present a keen shopping basket gadget based on minimal expense IoT equipment and profound learning object acknowledgment innovations. A camera for continuous item following, an ultrasonic sensor that fills in as an impetus, a weight sensor to choose when an item joins or leaves the shopping basket, an application that offers a UI for a virtual shopping basket, and a profound learning worker where learned item information are prepared make up the proposed shrewd truck system. TCP/IP and HTTP networks are utilized to speak with the modules, and the worker perceives the item utilizing the YOLO model darknet library, an article recognition strategy. The client will audit the rundown of items put in the savvy truck with a versatile application and pay naturally. The keen truck gadget proposed can be utilized to make minimal expense, elite automated stores.

Table of Contents

Chapter 1

Introduction

With the Internet of Things (IoT), gadgets and equipment can now interact over a wireless network infrastructure. Computing power and communication with common objects are now able to connect objects everywhere. They can be used to connect objects. As a result, a revolution is currently taking place in financial, environmental, and industrial systems, creating considerable hurdles in terms of data management, decision making and wireless communications in real time. In addition, several security and privacy issues have arisen, and encryption solutions, which are lightweight and are generally needed in IoT applications.

At present, the population of the world is growing rapidly, resulting in a wide range of needs in various areas. As human civilization evolves, supermarkets have become part of our daily lives, where consumers can use carts and baskets to look for the things they need. The things are placed on the trolleys and then brought to the check-out area. Although customers may acquire most of the commodities they need because of the availability in a wide range of products, finding items in an enormous supermarket can take time and lead to physical tiredness and mental dissatisfaction.

The Payment System that is currently being utilized by shopping and retail environments requires a ton of labour and time. As of now with the advanced deep learning techniques and modern technology automated stores that are transforming the shopping climate are surfacing.

One of the examples of the automated stores is that of the Amazon Go. In Amazon Go, there is no waiting time for the customers for bill generation. Automated stores are proven to be useful and provide an amazing customer experience as the wait time for customer's bill generation calculation and payment is diminished almost to zero. Thus there is expanding interest in this concept of smart unmanned stores here as well as in other parts of the world. [1].

Considering the Amazon Go which is introduced as a delegate model of automated stores, many cameras, receivers, and pressure sensors are utilized to track the customers continuously and to monitor contact among the customers as well as the products. From the point when a customer enters the store and scans his/her phone on the device installed at the entrance of the store, the customer is then considered as a 3D target of the system.

However, Amazon Go has a definitive shortcoming that is restricting the quantity of individuals in the store to no more than 100 because of AI reading issues. Furthermore, Amazon Go tracks and analyses the movement of each and every one of the customer which results in an enormous amount of information being gathered which renders it useless for huge distributing networks/stores.

In this paper, we propose a smart shopping trolley system that utilizes deep learning techniques for the purpose of object detection and Jetson Nano to reduce the disadvantages of the automated stores as have been discussed above. The proposed framework comprises of a camera, an ultrasonic sensor, TCP/IP-based networking system, deep learning server and an Android based smartphone application for the user.

In this smart shopping trolley system, the products can be added to or erased from the virtual shopping cart on the grounds that through this framework, the number, amount, and the type of product being put in the smart shopping trolley by the user can be communicated to the client's gadget continuously.

A few techniques for smart shopping trolley system that are being implemented include the perceiving of clients through face acknowledgment on the UI and using the RFID (Radio-Frequency Identification) labels to automatically identify different items added to the truck and show important data on the UI. Notwithstanding, appending RFID to all items requires cost and exertion.

However, in smart trolley system that we have proposed and implemented in this paper, it isn't important to have RFID labels attached to the products as since products are being recognized utilizing the cameras through object detection.

1.2. Motivation

We have seen long lines in the billing portion shopping marts that for most of the customers takes often takes more than time than then the time taken to shop all that they had shopped. While shopping purchasers deal with numerous issues like stressing that measure of cash brought isn't adequate, incomplete data about of the products that they require, other than this they need to choose the best item out of thousands of items they also have to deal with the stress of waiting in long queues just to bill their shopping products.

Thus for the sake of minimizing the billing to almost zero as well by automatic identification of products and automated billing not only the time taken for waiting to get their bills would be reduced but the problem of incomplete information regarding the products they require will also be solved.

Furthermore, a supermarket is a self-advantage shop offering a wide variety of sustenance and family figured out into various ways. Customers contribute a significant proportion of energy to find the things they require. If the Customers don't find the thing or the staff to assist them with excursion it is more likely that they leave the overall store with no purchase, which is believed to be an unprecedented disaster to the venders.

Disappointment imparted by customers because of long holding up time in the midst of the Checkout system is another genuine concern. Controlling the operational costs is totally perhaps the best trouble that any retailer faces. If work cost decreasing isn't directed suitably, client administration and store conditions may worsen. This obviously, bring about lost clients and deals.

Finally, another one of the reasons is the need to reform the whole shopping system in general as well as to reduce to the cost of money that must be paid to the labour that is required for billing purposes.

1.21. Why Object Detection?

Earlier RFID systems have been used in smart shopping carts however Object Detection seems to be much more useful and less costly as compared to the RFID system that is being implemented in some of the smart trolley systems as the components required for implementation for RFID is much more expensive.

Furthermore in case of an RFID system the customer would have to place the product at a specific angle in front of the RFID sensor for scanning the RFID tag while in case of object detection system that we have proposed customer would rather just put the product inside the cart regardless of the angle of the product and it would automatically be detected thus being more user friendly as well as less time consuming as the time for placing and scanning the RFID tag in front of the RFID scanner would be minimized.

1.3 Scope of the Project

Benefits

• Improve the shopping experience for all the customers of the store

• Increase efficiency of the exit process

• Eliminates a long waiting queues at the exit counter

Features

• User interface with LCD monitor for user inputs

• Automated shopping items detection system

• Automated communication system to make payments at counter or via app

• Automated data formatting in case of item deletion or additions and to organization the shopping in a systematic way.

1.4 Literature View:

One of the main challenges faced by customers when shopping in the store is the failure to find merchandise and even to transport goods to the billing counter. In the article [1], Author describes a new cost-effective approach to solve these problems by building a smart trolley using a web camera and video editing to complete the tasks. Compared to previous solutions using RFID transceivers, our solution costs 10 times less than its counterparts and is also environmentally safe.

Shelf Scanner [2], which aims to enable visually disabled people to shop in a grocery store without additional human assistance. Shelf Scanner makes it easy to identify objects on the shopping list online, in video streams in which any or all items can appear simultaneously. To cope with the size of the object detection mission, the machine leverages the estimated planarity of the grocery store shelf to create a mosaic in real time using an optical flow algorithm. The machine is then able to use any object detection algorithm without incurring data loss due to processing time. For speed purposes, they use a multiclass Naive-Bayes NIMBLE-inspired classifier trained on enhanced SURF descriptors derived from images in the GroZi-120 dataset. It is then used to measure the distribution of probabilities per class on video key - points for final classification. Their findings indicate that Shelf Scanner may be useful in cases where high-quality training data is available.

The Article [3] introduces a new SURF-coated detector and descriptor, invariant in scale and rotation (Speeded-Up Robust Features). SURF is approximating or even outperforming previously proposed schemes in terms of repeatability, distinctiveness and robustness, but can be calculated and compared much faster. This is achieved by relying on integrated images for image transformations; building on the abilities of the top current detectors and identifiers (specifically, using a Hessian matrix-based measure for the detector and a utilization descriptor); and simplifying these methods to the essentials. This results in a combination of novel detection, description and matching steps. The paper includes a detailed description of the detector and descriptor and then describes the effects of the most important parameters. With the application of SURF, this conclude the article with two challenging but converse objectives: camera calibration as a special case of image registration, and object recognition. Their experiments underline SURF's usefulness in a broad range of topics in computer vision.

The large-scale image recovery instance-level is intended to retrieve certain object or scene instances. At the same time, the retrieval of many items in a test image adds to the issue especially if they are visually similar. This document [4] provides a successful solution per exemplary multilabel image recognition, which helps to find and localize goods in retail images. The use of discriminatory random forests, deformable dense pixels and genetic optimization allows to achieve runtime efficiency. Cross-dataset recognition is carried out where our training images are ideally taken with just one training picture per product label, while the evaluation set is carried out in entirely different real-life scenarios using a mobile phone. In addition, its offer a large range of new data sets and labelling tools for image search products to further investigate the image classification of multi-label retail products. The proposed approach achieves good results in terms of precision and performance for 680 of our dataset's annotated images and 885 of GroZi-120's test images. They have 8350 different product pictures and 680 retail test pictures. Shops with complete annotations for the wider community.

This article [5], suggest a SIFT and KAZE accessibility main point collection scheme and show its utility in object characterization. The selection criteria depend on the detectability, features of key points and their repeatability. These scores are combined to provide a main point production. The key points are ranked by their selling prices and a threshold is used to weed out the weak/irrelevant key points. The key points obtained by using SIFT to the structure map generated by Gabor filter are further enhanced. The keyboard collection effectively represents the frontiers and object zones. Experimental findings validate the argument that the key points selected for object representation by the technique suggested are well-suited.

A rigorous methodology for object identification and recognition; Words Bag and Deep Neural Networks are compared. A step-in post-processing to integrates multiple detection of the same object. A detailed experimental assessment on the dynamic public data set of Grozi-120. Detection and identification of objects is complicated due to the vast number of implementations by computer vision activities. This work [6] concentrates on items in store shelves, offering supplementary product/price knowledge to the consumer or advising visually disabled shoppers during shopping in a variety of practical applications. The emphasis is This structure. Automatic planograms (that is, the real shelf arrangement of products) is also ideal for the market research and operation of large shops. This paper show that our selective search enables the use of the powerful Bag-of-Words model for recognition.

Recent methods for texture and object recognition tasks based on local image features showed promise. A large-scale evaluation approach is presented in this paper [7], which represents images as a distribution (signatures or histogram) of features from a small number of locations, and learns a Support Vector Classification Machine with kernels based on two effective distribution comparison measures, the Earth mover and the ·2. Initially, our approach evaluates the performance of various key point detectors, descriptors and kernels and classifications. They then conduct a comparative assessment of four texture and five object databases using several state-of-the-art recognition methods. Our implementation exceeds the best reported results in most of these databases and results in comparable results for the rest.

For an electronic payment system with limited human intervention, 3D object recognition systems may be very useful. Many algorithms have been programmed to identify artefacts that differ in output, precision, time of computation, etc. A realistic computer vision system for electronic payment systems must be extremely easy and accurate, as a mistake will lead to an incorrect payment with a significant economic impact. SIFT is a high precision but high memory requirement floating point descriptor. PCA-SIFT is a small-scale, adapted variant of SIFT. FREAK is a binary descriptor that takes relatively little buffer space, less extraction, and less time than SIFT. A hybrid PCA-SIFT-FREAK algorithm that aims to achieve a compromise between precision and memory is proposed in the current work. The output parameters such as recognition rate, measurement time, memory specifications and SIFT, PCA-SIFT and Hybrid confusion matrix are compared [8].

1.5 Structure

The report is structured in the following way:

- Chapter-2 mainly deals with the design of the Smart Trolley System as to what components have been chosen and who will they be combined together to achieve the final design.
- Chapter-3 deals mainly with integration of the design components as well their internal functioning also deals with the training procedures that the Yolo v4 model has to go through and the internal deep learning techniques involved in the whole of the training process.
- Chapter-4 deals with hardware implementation of the proposed system.
- Chapter-5 Covers details about Database, GUI and their Connection
- Chapter-6 includes Business Plan

Chapter 2

Smart Trolley System Design

2.1 Design of the Smart Trolley System

The goal of the work is to provide a pleasant shopping experience for the customers. The product information of all store merchandises and reducing delay billing time are mainly considered in shopping basket implementation. Customers face difficulties of wasting time at the billing counter after shopping and getting information about the products they want to purchase. Continuous improvement to the automatic billing system is needed to improve the quality of shopping practice to the customers. So, the design of the proposed shopping basket can be shown in Fig. 1 and fig. 2

The smart shopping basket combines a shopping basket with a camera mounted on top of the basket. It benefits the customer. By self-identifying the purchased things that each customer places in their shopping basket. If the consumer wishes to delete products that can be performed by scanning the goods from the cart again. Here the detected things are displayed using an LCD. As soon as a customer logs in, they are granted a basket id, which they will use throughout their purchasing experience. The smart shopping trolley comprises of an ultrasonic sensor that identifies that the product is being placed into the trolley, pie cameras that detects the products, and a Jetson Nano that controls the whole framework and performs network correspondence.

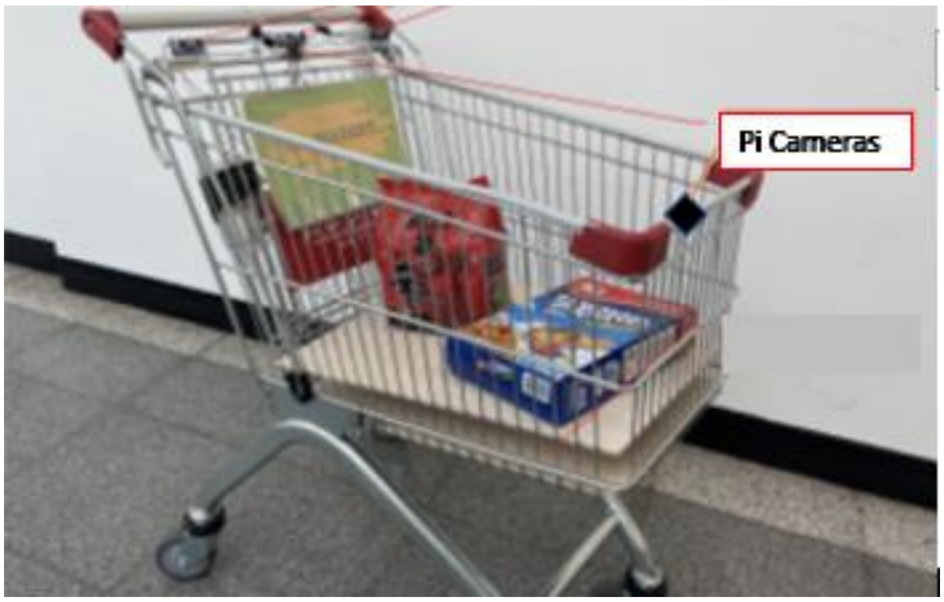


Fig 1. Front View of the Proposed Smart Trolley.

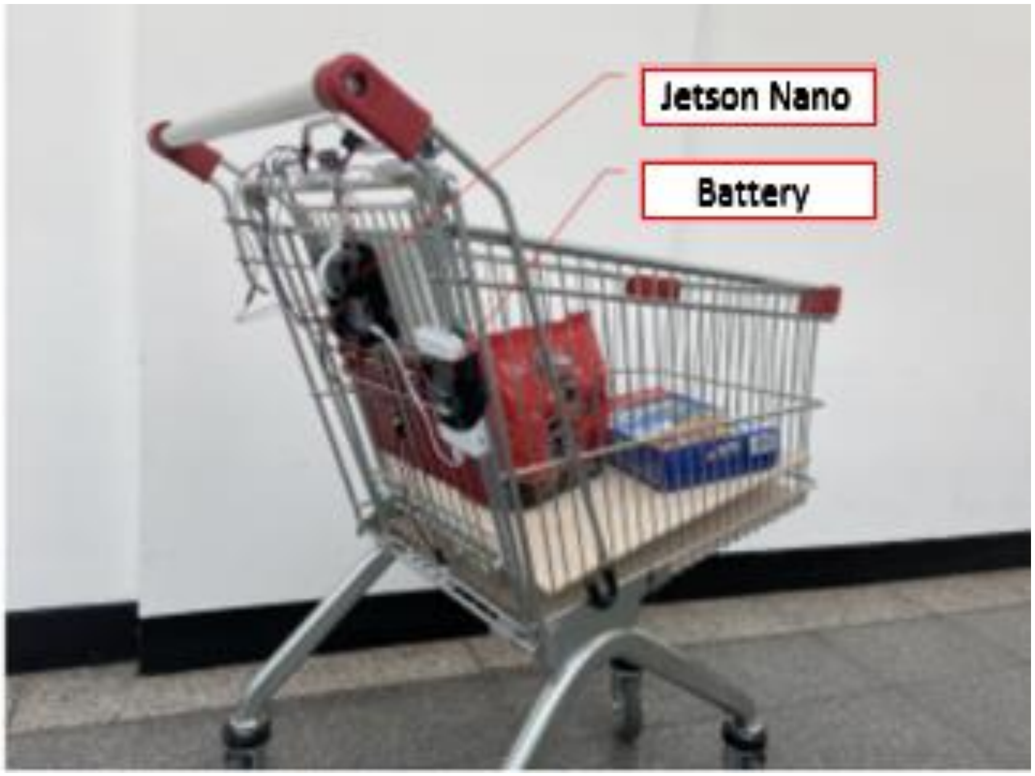


Fig 2. Back View of the Proposed Smart Trolley.

The scanned goods are automatically invoiced in the Desktop software, cutting turnaround time dramatically. The scanned items are also sent to the shop's central server. Using this approach, the time-consuming task of scanning and charging each and every good at the cash register may be avoided. A weight sensor can also be incorporated at the bottom of the shopping basket. It just checks to see whether any products are added without being scanned, so that the increased weight in the basket may be detected.

The required system must be dependable while scanning items, consistent in delivering appropriate replies to operations, and capable of appropriately transmitting all info to an online database. We propose a smart trolley system in which customers scan items and finish the invoicing procedure within the trolley itself.

Fig. 3 below shows the segments and interfaces that make up the proposed brilliant shopping basket framework.

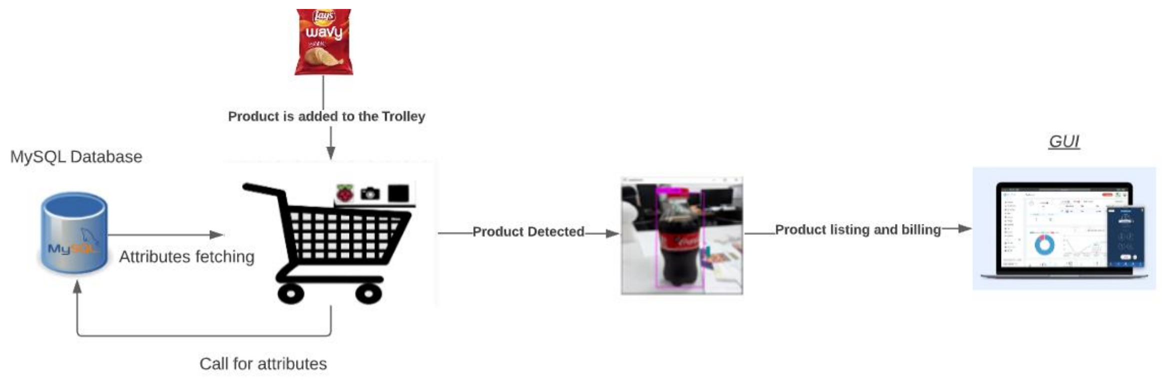


Fig 3. Schematic Diagram for Smart Shopping Trolley.

2.2 Design Requirements

When designing the smart shopping trolley, the following design requirements must be taken into consideration:

- For accommodation of shopping, it ought to be worked with a remote battery.
- Minimal power ought to be used for conservation of battery power.
- It should be recognizable even if multiple products are simultaneously being put in or taken out of the trolley.
- It should be distinguishable whether a product is being put inside the trolley or if a product is being taken out of the trolley.

2.3 Order of Operation

The Shopping begins when the user login the shopping cart with his/her credentials. After the Login, the Camera attached with the Jetson Nano in the shopping cart recognizes the product when the user puts or pulls out the product. The YOLO v4 Model deployed on Jetson Nano Identifies the product and predict the label of the product. The predicted label is Used to run a Query which fetches All the info (weight, price, Quantity) related to the product.

Upon receiving the query response, the app adds product data in the virtual shopping cart. When shopping is completed, the user pays with his or her smartphone, returns the shopping cart to its original location and exits the mart while closing their shopping. Then the shopping cart begins charging.

2.4 Software & Hardware Used:

Software:

Google Colab:

An online IDE for Python that enables Machine Learning with cloud storage. Colaboratory, or "Colab" for short, is a Google Research product. Colab enables anybody to create and execute arbitrary Python code via the internet, and it is particularly suitable to machine learning, data analysis, and education.

Anaconda:

Anaconda is a Python and R programming language distribution aimed for simplifying package management and deployment in scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, and so on). Data-science software for Windows, Linux, and macOS are included in the release.

MySql:

MySQL is a relational database management system that is free and open source (RDBMS). A relational database organizes data into one or more data tables where data types can be associated to one another; these relationships assist structure the data. SQL is a programming language that allows programmers to build, change, and retrieve data from relational databases, as well as control user access to the databases. In addition to relational databases and SQL, an RDBMS such as MySQL collaborates with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access, and simplifies verifying database integrity and backup creation.

Hardware:

RASPBERRY PI 4 MODEL B 2GB:

The Raspberry Pi 4 model B was launched with an ARM Cortex-A72 CPU with a 1.5 GHz 64-bit core 64 GHz, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet, 2 USB 2.0, 2 USB 3.0 port and two HDMI (HDMI Type D) supports with a resolution of up to 4K. The Pi 4 is also powered through a USB-C connection, so that additional power may be supplied with the necessary PSU to downstream devices. We are using 2 GB model.

Rpi Camera:

The Raspberry Pi Camera Module v2 is a custom made add-on board for Raspberry Pi with an 8 megapixel Sony IMX219 image sensor and a fixed focus lens. It has a static picture resolution of 3280 x 2464 pixels and video resolutions of 1080p30, 720p60, and 640x480p90.

Specs	Component	Model
Hardware	Nvidia Jetson Nano 2GB	Jetson Nano 2GB developer kit
	PiCamera	PiCamera v2
	Python	Python 3.7
Object Detection model	YOLO	Yolov4 Darknet
Database	MySQL	MySQL community server 8.0.25

2.5 Dataset:

We Collected Dataset of about 19 Classes Manually as shown below:

- Colgate Toothpaste
- Elbow Macroni
- Lasagne
- Mitchell's Strawberry Jam
- Olper’s
- Oreo Mini
- Perk double
- Power Plus
- Shan Korma Masala
- National Achar Gosht
- Spaghetti (Bake parlour)
- Tapal Green Tea (Elaichi)
- Whistlez Biscuit
- Italia Corn Flour

- Marlboro
- Rafhan Pudding
- Robin Bleach

2.5.1 Annotation:

The dataset is then manually annotated for Csv file as it is needed by the Model to predict the Product. Makesense.Ai is used for this purpose. Makesense.Ai is an online photo labelling tool that is completely free to use. It does not require any difficult installation due to the usage of a browser - simply visit the website and you are ready to begin. It also makes no difference what operating system you're using.

Chapter 3

Yolo v4(Architecture & Model Training)

3.1 Implementation of the Smart Trolley System

The smart shopping trolley system has a Jetson Nano 2GB. It is powered by an adapter of specs 5v-3A. We used a Pi Camera to accept the video. RPi camera is Attached with the Jetson Nano. Live feed from the RPi Camera is fed to our Model Which in turn detects the object and add it to the shopping list in the billing Cart which is a GUI made on Tkinter.

3.2 Object Detection

In a smart shopping trolley, it is a vital undertaking to accurately detect the products added to the trolley. Subsequently to getting the image, through the camera, the shopping trolley ought to recognize the product in the image and accurately find out the location of the product of the art. figure out the area of the article. We also need to consider the possibility about the circumstance in which the client places a few items into the trolley.

The suggested smart cart system utilizes a YOLO library which provides one of the deep learning algorithms (you only look once, henceforth YOLO). YOLO is an open-source package which not only provides a classifier but also techniques to locate numerous items inside a picture.

The YOLO library offers superior performance than prior neural networks for object detection, a robust real-time object detection system which supports real-time video detection through a camera and file format video. Since YOLO is based on CNN, it provides a convolution-based architecture, which samples the input picture size, such as the convolutionary layer and sub-sampling.

YOLO predicts a number of bounding boxes around the picture using a single CNN and utilizes an integrated model to compute the class probability in each box simultaneously. YOLO is almost 1,000 times quicker than conventional R-CNN, 100 times faster than Fast R-CNN, and 10 times faster than the last Faster R-CNN.

Since YOLO also handles the problem using supervised learning, high-quality, well-labelled data must be secured. When the object detection problem occurs, the right response label consists of a pair and an annotation of the label name of each item.

Only 10 product classes were considered in this study, while 150-200 learning data were created utilizing cameras for each class, such as resizing and marking of 1400 learning data utilizing the Makesense.ai annotation tools. Moreover, the version of YOLO we used in our project is YOLOv4.

Scores are calculated for each bounding box and Output only the bounding boxes above the threshold as shown in fig 6 below:

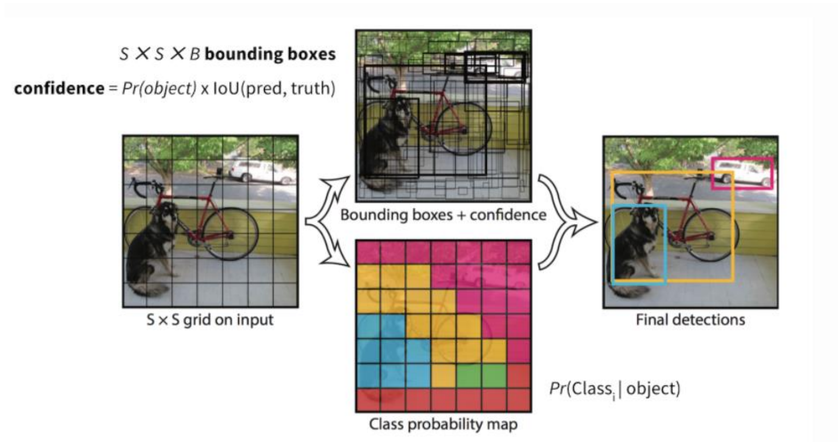


Fig 6. Bounding Boxes & Score of Each Object.

3.3 Yolo v4 Model Internal Architecture

3.3.1 Architecture:

An object detector basically consists of two parts that are: a backbone (pre-trained on ImageNet) and a head (for predicting the classes and the bounding boxes of the objects). However, in case YOLO some layers have been insert between the backbone and the head for the purpose of collecting the feature maps from different stages. This is known as the neck of the object detector. [14]

In case of YOLOV4 the architecture consists of CSPDarknet53[10] as its backbone, spatial pooling additional module (SPP) [11] and PANet path aggregation (PAN) [12] as its neck with YOLOv3[13] as its head.

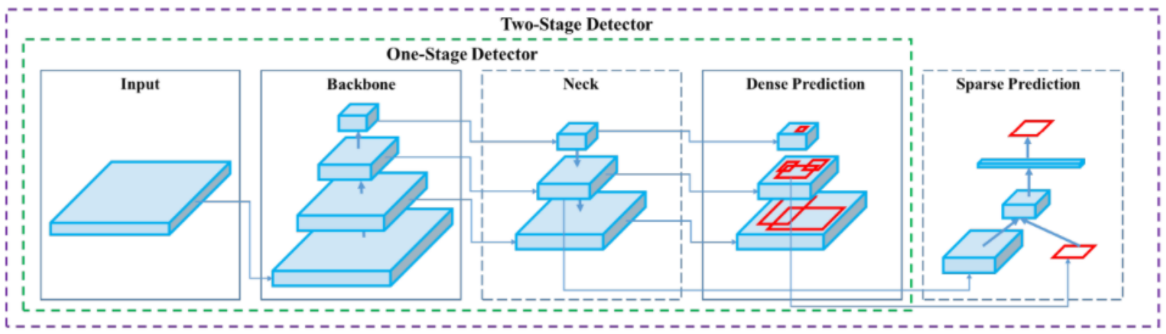


Fig 7. Architecture of Yolo

The following is breakdown of the architecture of YOLOv4:

- Backbone: CSPDarknet53[10]
- Neck: SPP [11], PAN [12]
- Head: YOLOv3 [13]

3.3.2 Explanation for choosing this Architecture Set for YOLOv4:

The CSPResNext50 contains only 16 convolutional layers 3 × 3, a 425 × 425 receptive field and 20.6 M parameters, while CSPDarknet53 contains 29 convolutional layers 3 × 3, a 725 × 725 receptive field and 27.6 M parameters. This theoretical justification, together with our numerous experiments, show that CSPDarknet53 neural network is the optimal model of the two as the backbone for a detector [10].

The purpose of adding the SPP block over the CSPDarknet53 is to increase the receptive field as well as to separate out the most significant context features. Moreover, PANet has been used as the method of parameter aggregation from different backbone levels for different detector levels. [10].

Yolov4 Model Uses:

Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing [9].

Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC) [9].

Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, eliminate grid sensitivity, using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes [9].

Bag of Specials (BoS) for detector: Mish activation, SPP-block, SA [9]

Bag of Freebies:

Bag of Freebies is a collection of approaches or strategies for improving model accuracy by changing the training approach or cost.

The Bag of Freebies in Object Detection is implemented in the following ways:

- Data Augmentation
- Semantic Distribution Bias in Datasets
- Objective Function of Bounding Box Regression

Data Augmentation:

Data Augmentation is basically the technique of increasing the heterogeneity in the input images of data so that the designed object identification model is more resistant to photographs from various contexts [15]. The figure below shows an illustration of Data Augmentation:

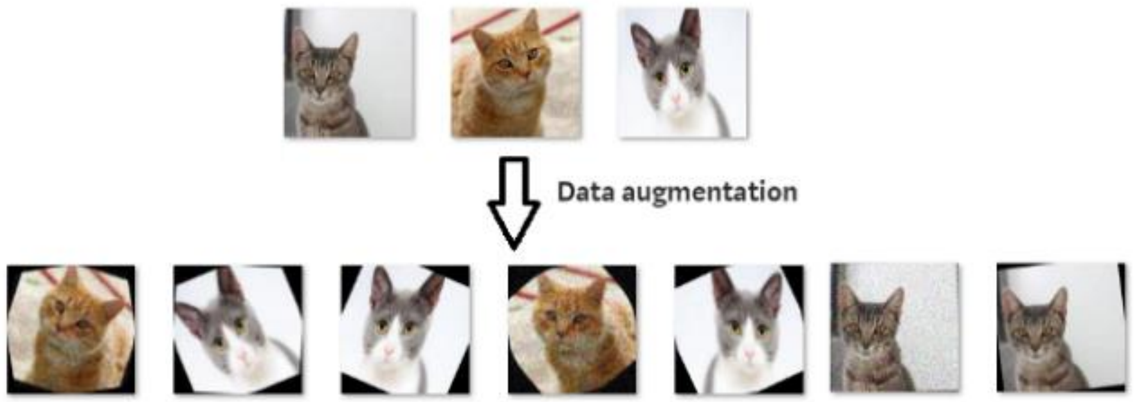


Fig 8. Data Augmentation

Semantic Distribution Bias in Datasets:

When it comes to the diversity of datasets on which the models are trained, Semantic Distribution Bias is a major challenge. If there is a bias in the data distribution, this might push the training towards a non-optimal convergence and failure to generalize. The causes for Semantic Distribution Bias are as follows:

- Uneven distribution of data among different classes
- Failure to express the degree of association between several categories.

Objective Function of Bounding Box Regression:

The objective function, also known as loss functions, is used in object detectors to penalize and drive the model towards improved convergence at each training phase.

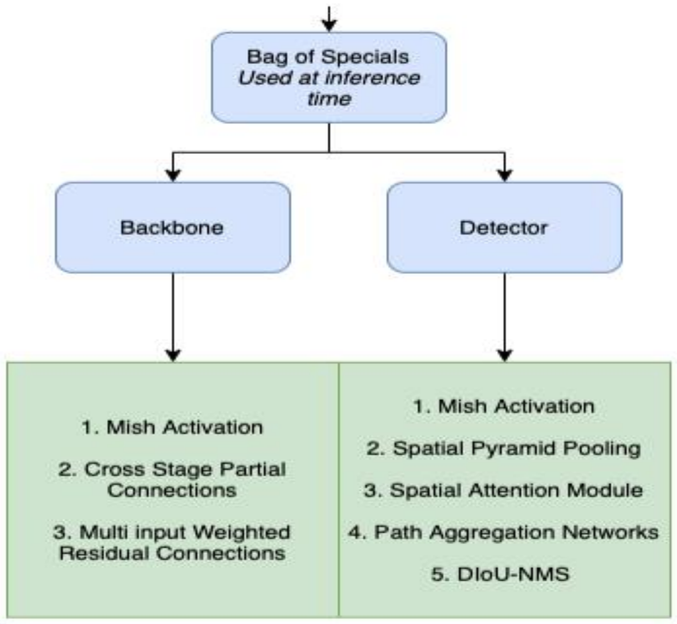
Bag of Specials:

Bag of Specials comprises many plugins and post-processing modules that only add a little amount to the inference cost but vastly improve the object detector's accuracy.

In general, these approaches, also known as Bag of Specials, may be used as an add-on to any object detector currently available to improve accuracy on benchmark datasets. Furthermore, the approaches chosen may differ depending on the design of the Object Detector, but the end aim of improving the detector outputs will be fulfilled.

Bag of Specials consist of the following:

- Mish Activation
- CSP (Cross Stage Partial Connections)
- FCN-Spatial Pyramid Pooling
- Spatial Attention Module
- Path Aggregation Networks (PANet)
- DIoU NMS


Bag of Specials
Used at inference time
Backbone
Detector
1. Mish Activation
2. Cross Stage Partial Connections
3. Multi input Weighted Residual Connections
1. Mish Activation
2. Spatial Pyramid Pooling
3. Spatial Attention Module
4. Path Aggregation Networks
5. DIoU-NMS

3.3.3 Mish Activation

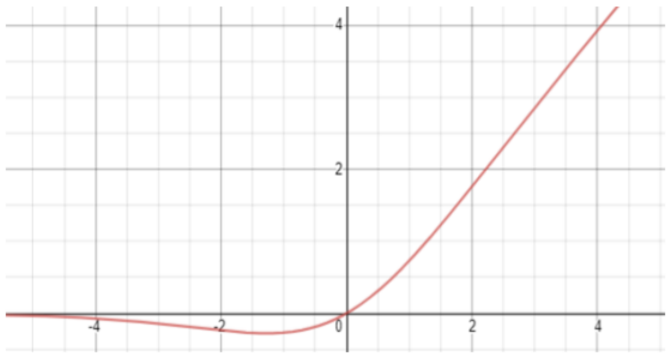
Mish is a novel activation function like Swish and is defined as:

$$f(x) = x \cdot tanh(softplus(x)) = x \cdot tanh(\ln(1 + e^x))$$

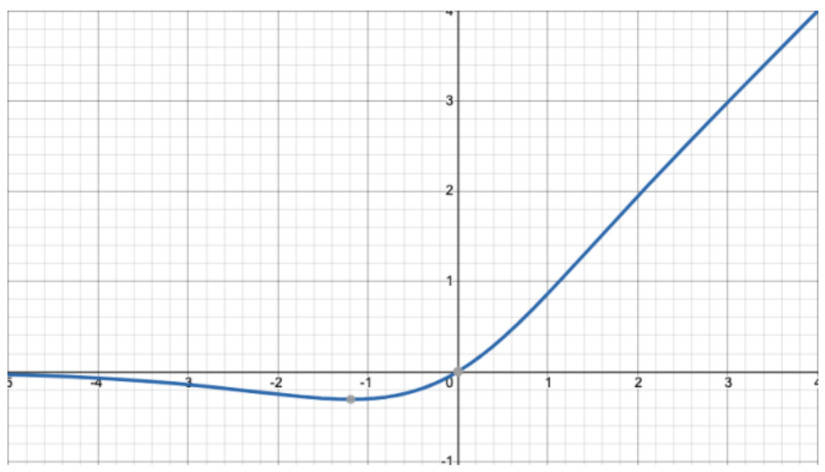
f(x) = x · tanh(ς(x))

where, **ς(x) = ln(1+e^x)**, is a **softmax** activation function.

This is very similar to another activation function called Swish function, that can be defined as:



$$f(x) = x \cdot sigmoid(x)$$



$f(x) = x \cdot tanh(\varsigma(x))$

The reason why Mish function is used in YOLOv4 is because of its low cost and its various properties like it's smooth and non-monotonic nature, unbounded above, bounded below property improves its performance when compared with other popularly used functions like ReLU (Rectified Linear Unit) and Swish.

3.3.4 CSP (Cross Stage Partial Connections)

The architecture of the cross stage partial derives from DenseNet, which utilizes the above input and links it with the current input before proceeding into the dense layer.

There is a dense block and transition layer on every layer of stage of a DenseNet and a dense block is each made from k dense layers. The output of this dense layer is linked to the input of the dense ith layer and the combined result becomes the input of the dense I + 1) layer.

3.3.5 FCN-Spatial Pyramid Pooling

The resulting feature map is flattened and routed to the FC layer for additional SoftMax action during classification jobs. However, in order to employ the FC layer, we must set the size of an input picture when training, which makes it difficult to recognize objects at various sizes and aspect ratios as shown in figure below:

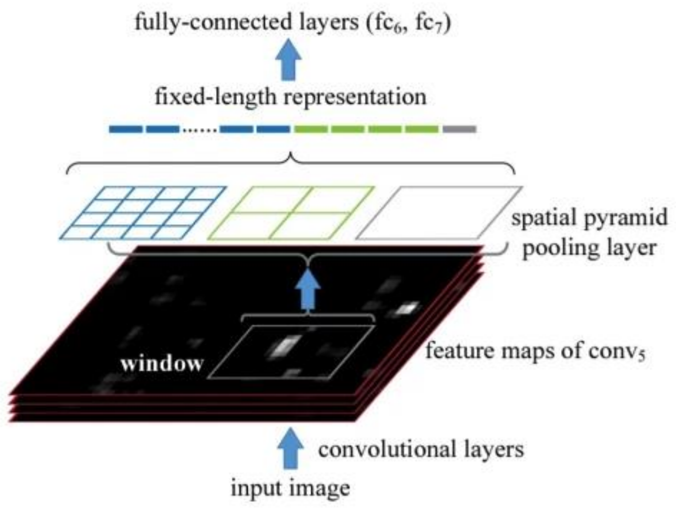

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
window
feature maps of $conv_5$
convolutional layers
input image

Fig 9. Spatial Pyramid Pooling

To address this problem, the final output feature map is subjected to channel-wise pooling for various spatial bin sizes. If the dimensions of the input feature map are 512X100X100(CXHXW) and the spatial bins are 1X1, 2X2, 4X4, SPP creates 512, 4*512, and 16*512 1-D vectors, which are then concatenated to fed into the FC layer.

Figure Below shows how SPP module is integrated.

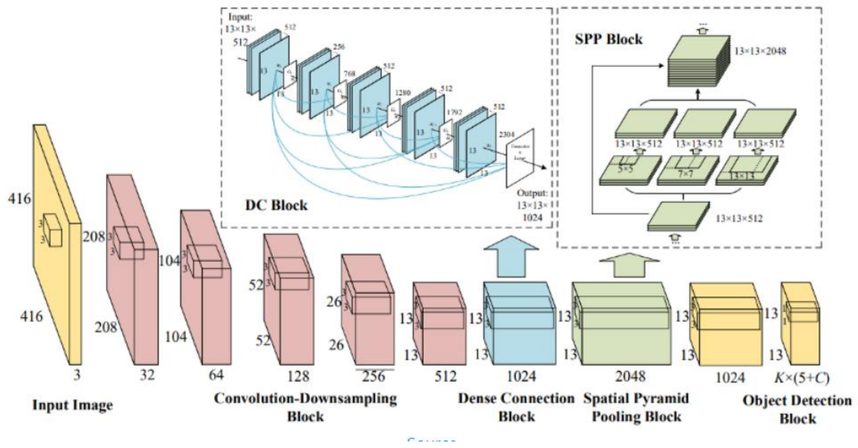


Fig 10. SPP Module

3.3.6 Spatial Attention Module

Attention modules have recently been employed in Convolutional Neural Nets to make the network focus on the objects present in an image rather than the entire image. These modules assist in answering the where and what questions by instructing a network to place greater emphasis on contextual information around an item and which traits are most essential, respectively. The Special Attention Module (SAM) is a spatial-wise attention layer that is commonly employed in CNNs.

3.3.7 Path Aggregation Networks (PANet)

During the early days of deep learning, a series of layers were utilized for rudimentary networks. From the preceding layer each layer is entered. The early layers capture localized texture and pattern information to create the later layers of semantic information. However, as we move towards the right, localized knowledge may be lost to adjust the prediction.

PaNet has created an architecture to improve the spread of layer data from top to bottom. To fix this problem. Typically, the neck components flow up and down between layers and link at the conclusion of the convolutionary network to the few levels.

The current layer and information on an earlier layer are added to generate a new vector when PaNet was originally implemented. A modified version is utilized for the YoloV4 execution to generate the new vector by connecting the input and the vector from the preceding layer.

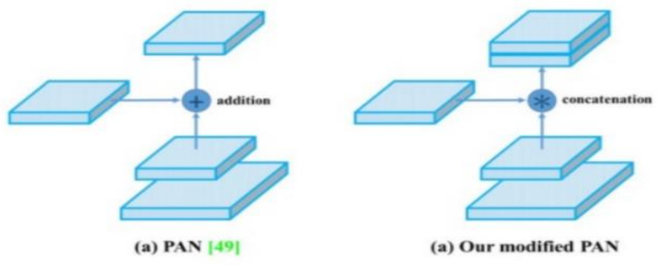


Fig 11. PANet

3.3.8 DIoU NMS

Based on the IoU metric system, NMS is used to remove unnecessary boxes. Due of the confidence level, two overlapping distinct boxes are frequently suppressed to one box when employing this approach.

3.3.9 How YOLOv4 Architecture is better than the older versions:

The main difference between YOLOv3 and YOLOv4 is that YOLOv3 has Darknet53 as the backbone and yolov4 has CSPDarknet53 which has 29 convolutional layers 3*3, a 725*725 receptive field and 27.6M parameters which is the optimal model of as the backbone for the detector based on theoretical justification and experiments [9]. Moreover, YOLOv3 model Feature Pyramid Networks are used for object detection YOLOv4 uses PANet is used as the method parameter aggregation for different detector levels.

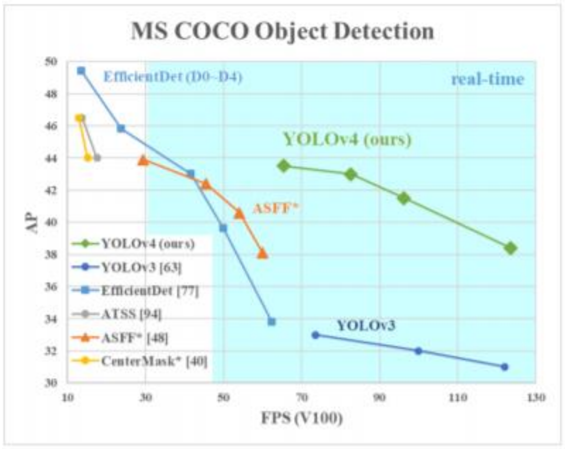
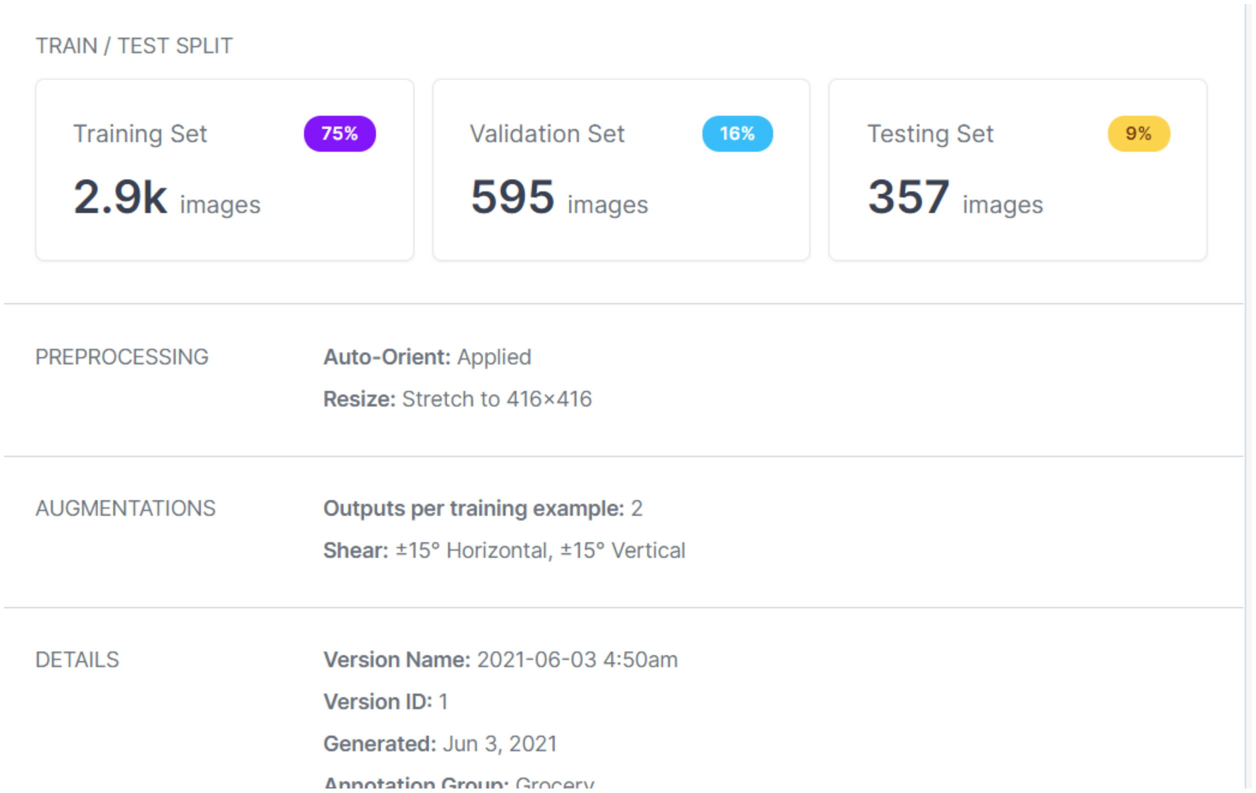


Fig 12. MS COCO Object Detection

3.4 Model Training & Evaluation:

The YOLO neural network learning rate has been 0.001 and 0.949 momentum and a weight decrease has been set at 0.0005. To train our model 10000 epochs were trained and the ultimate loss function value was found to be 0.0646. We set the threshold to the default value 0.25 in the prototype model to lower the rate of insertion of the incorrect product.


TRAIN / TEST SPLIT

Training Set 75%
2.9k images

Validation Set 16%
595 images

Testing Set 9%
357 images

PREPROCESSING
Auto-Orient: Applied
Resize: Stretch to 416×416

AUGMENTATIONS
Outputs per training example: 2
Shear: ±15° Horizontal, ±15° Vertical

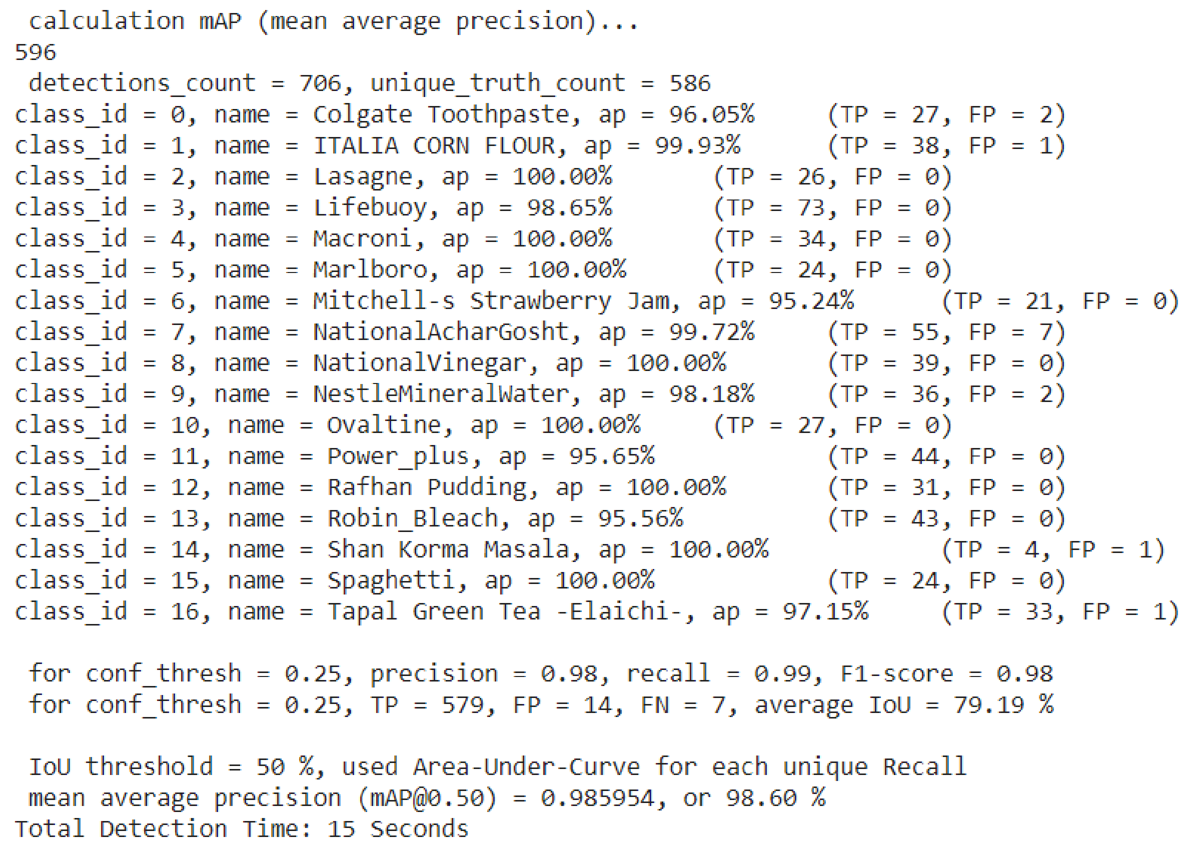
DETAILS
Version Name: 2021-06-03 4:50am
Version ID: 1
Generated: Jun 3, 2021
Annotation Group: Grocery

Figure Training and splitting of dataset

Once the learning is complete, real-time measurements may be tested with a file that keeps the weight learnt. The 17 goods have been evaluated, including: Shan Karahi Masala, Shan Qorma Masala, Marlboro, Tapal Green Tea etc. Data is collected and 39 detections per trigger are collected in real time. The average accuracy of detection is determined and recorded in a round for this 39 detection result.

This method was conducted ten times per product class, and the average system prediction values for each class were derived based on the mean average (mAP@.50) accuracy of the entire system. The system map was calculated at 98.60%.

3.5 Training Results:

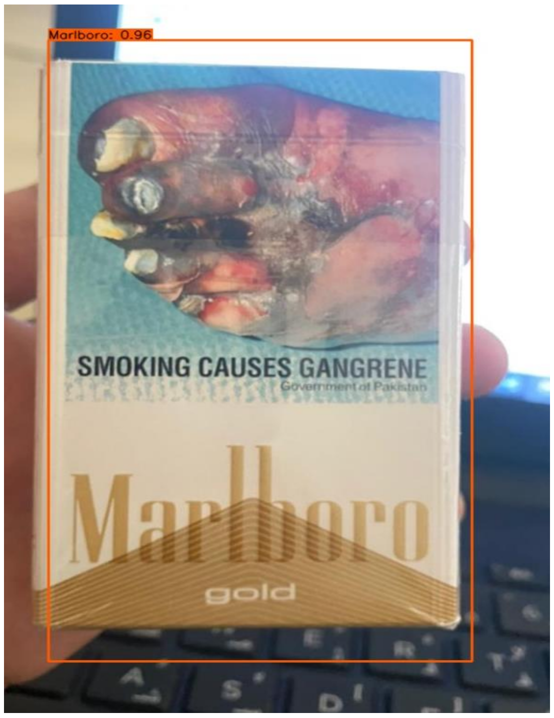


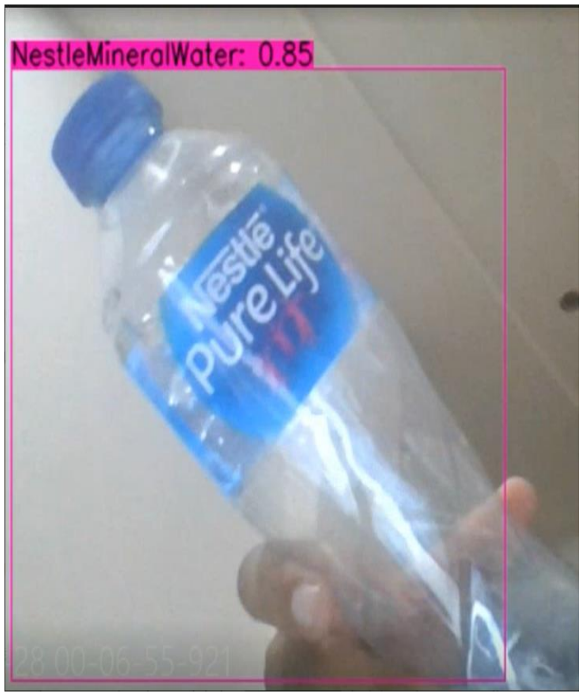
```
 calculation mAP (mean average precision)...
596
 detections_count = 706, unique_truth_count = 586
class_id = 0, name = Colgate Toothpaste, ap = 96.05%     (TP = 27, FP = 2)
class_id = 1, name = ITALIA CORN FLOUR, ap = 99.93%      (TP = 38, FP = 1)
class_id = 2, name = Lasagne, ap = 100.00%       (TP = 26, FP = 0)
class_id = 3, name = Lifebuoy, ap = 98.65%       (TP = 73, FP = 0)
class_id = 4, name = Macroni, ap = 100.00%       (TP = 34, FP = 0)
class_id = 5, name = Marlboro, ap = 100.00%      (TP = 24, FP = 0)
class_id = 6, name = Mitchell-s Strawberry Jam, ap = 95.24%      (TP = 21, FP = 0)
class_id = 7, name = NationalAcharGosht, ap = 99.72%     (TP = 55, FP = 7)
class_id = 8, name = NationalVinegar, ap = 100.00%       (TP = 39, FP = 0)
class_id = 9, name = NestleMineralWater, ap = 98.18%     (TP = 36, FP = 2)
class_id = 10, name = Ovaltine, ap = 100.00%     (TP = 27, FP = 0)
class_id = 11, name = Power_plus, ap = 95.65%            (TP = 44, FP = 0)
class_id = 12, name = Rafhan Pudding, ap = 100.00%       (TP = 31, FP = 0)
class_id = 13, name = Robin_Bleach, ap = 95.56%          (TP = 43, FP = 0)
class_id = 14, name = Shan Korma Masala, ap = 100.00%            (TP = 4, FP = 1)
class_id = 15, name = Spaghetti, ap = 100.00%            (TP = 24, FP = 0)
class_id = 16, name = Tapal Green Tea -Elaichi-, ap = 97.15%     (TP = 33, FP = 1)

 for conf_thresh = 0.25, precision = 0.98, recall = 0.99, F1-score = 0.98
 for conf_thresh = 0.25, TP = 579, FP = 14, FN = 7, average IoU = 79.19 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.985954, or 98.60 %
Total Detection Time: 15 Seconds
```

Fig 13. Confusion Matrix

As we can see it is evident from our confusion matrix that our model is less accurate in identifying the fruit related products.

Result:


Marlboro: 0.96
SMOKING CAUSES GANGRENE
Government of Pakistan
Marlboro
gold


NestleMineralWater: 0.85
Nestlé Pure Life

CHAPTER 4

HARDWARE IMPLEMENTATION

4.1 H/W used:

- RASPBERRY PI 4 MODEL B
- RPI camera v2 8 Megapixel
- 5V-3A Power Supply
- LCD
- Mouse and Keyboard

4.2 Implementation:

The Raspberry Pi 4 model B was launched with an ARM Cortex-A72 CPU with a 1.5 GHz 64-bit core 64 GHz, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet, 2 USB 2.0, 2 USB 3.0 port and two HDMI (HDMI Type D) supports with a resolution of up to 4K. The Pi 4 is also powered through a USB-C connection, so that additional power may be supplied with the necessary PSU to downstream devices. We are using 2 GB model.

4.2.1 Setting up Raspberry Pi:

When you initially take the Raspberry Pi 4 out of the box, it's nothing but a circuit board with tons of potential but none of the ready-to-go capabilities of a conventional PC or laptop. The operating system is not preloaded; you must install it yourself, and any peripherals, such as a keyboard, mouse, or monitor, must be purchased separately.

Even with your peripherals in place, you'll need to install the program manually. To begin, obtain a microSD card and download Raspbian Buster, the operating system created exclusively for the Pi 4 by the Raspberry Pi Foundation (though it will also work with past models). Raspbian Buster, the most recent version of the Raspberry Pi-themed Linux derivative, has been upgraded to match the new Pi 4's improved capabilities and features. Raspbian may be downloaded straight from the Raspberry Pi Foundation. Download the official Buster operating system image and save it to your microSD card.

Once the Raspbian Buster image is on the microSD card, slide it into the card slot on the Pi 4 and power up the Pi, which will immediately boot into the newly installed OS provided everything was loaded successfully. While these instructions will get you started, Raspbian Buster isn't the only operating system for your Raspberry Pi. Several Linux-based operating systems are publicly available for the RasPi, however their compatibility with the Pi 4 may be more difficult to establish.

Using Raspbian Buster on the Raspberry Pi 4 provides a normal desktop experience, albeit a rudimentary, Linux-based one. A suite of programs that provide basic online surfing and productivity capabilities, as well as a selection of educational apps that teach coding and provide tools for customizing the Pi to your own unique projects, are included in the initial installation package.

4.2.2 Setting up Camera Module:

All current Raspberry Pi models include a connector for attaching the Camera Module. The Camera Module (RPI camera v2) is available in two variants:

The regular model, which is intended for use in typical lighting conditions.

The NoIR version, which lacks an infrared filter and may thus be used in conjunction with an infrared light source to capture images in the dark.

First Check that your Raspberry Pi is turned off. Identify the Camera Module port. Carefully pull up on the dock's plastic clip's edges. Connect the Camera Module ribbon cable, making sure it is the correct way around. Return the plastic clip to its original position. Begin by turning on your Raspberry Pi. Open the Raspberry Pi Configuration tool from the main menu. Select the Interfaces tab and make sure the camera is turned on. Restart the Raspberry Pi. If all Goes well your camera will be up and running.

4.2.3 Installing Dependencies:

Following Dependencies must be completed before deploying our model on Raspberry Pi 4.

- **OpenCV=4.1.1.26**
- **Tensorflow ==2.3.0rc0**
- Lxml
- Tqdm
- absl-py
- Easydict
- Pillow

OpenCV and Tensorflow:

Both Open CV and Tensorflow have to be installed by source as we have to use the specified versions.

For **OPEN CV.** The scripts below will update and upgrade any existing packages before installing OpenCV dependencies, I/O libraries, and optimization packages:

The first step is to update and upgrade any previously installed packages:

```
sudo apt-get update && sudo apt-get upgrade
```

After that, we'll need to install various development tools, including CMake, which will assist us in configuring the OpenCV build process:

```
sudo apt-get install build-essential cmake pkg-config
```

Following that, we must install certain image I/O packages that will enable us to load multiple image file types from disk. JPEG, PNG, TIFF, and other similar file formats are examples:

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
```

Video I/O packages are required in the same way that image I/O packages are. These libraries enable us to read multiple video file types from disk as well as directly interact with video streams.

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
```

The OpenCV library includes a sub-module called highgui, that is used to show pictures on our screen and construct simple GUIs. In order to compile the highgui module, we must first install the GTK development library and the following prerequisites:

```
sudo apt-get install libfontconfig1-dev libcairo2-dev
sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

Many OpenCV operations (particularly matrix operations) may be significantly improved by adding a few additional dependencies:

```
sudo apt-get install libatlas-base-dev gfortran
```

Finally, install Python 3 header files so that we may compile OpenCV with Python bindings:

```
sudo apt-get install python3-dev
```

Create your Python virtual environment and install NumPy

In dealing with Python, we will be employing virtual environments from Python, a solid practice.

A virtual environment is separated from your system – it is completely sequestered from other surroundings. In addition, you may use pip (Python package manager) to handle Python packages inside your virtual environment.

Alternatives to manage virtual environments and packages (notably Anaconda/conda) are of course available. We have used / trialed them all, but have established the favorite tools I install on all my systems: pip, virtualenv and virtualenVwrapper.

Pip can be installed by using following command:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo python3 get-pip.py
sudo rm -rf ~/.cache/pip
```

Let's Install virtualenv and virtualenvwrapper now:

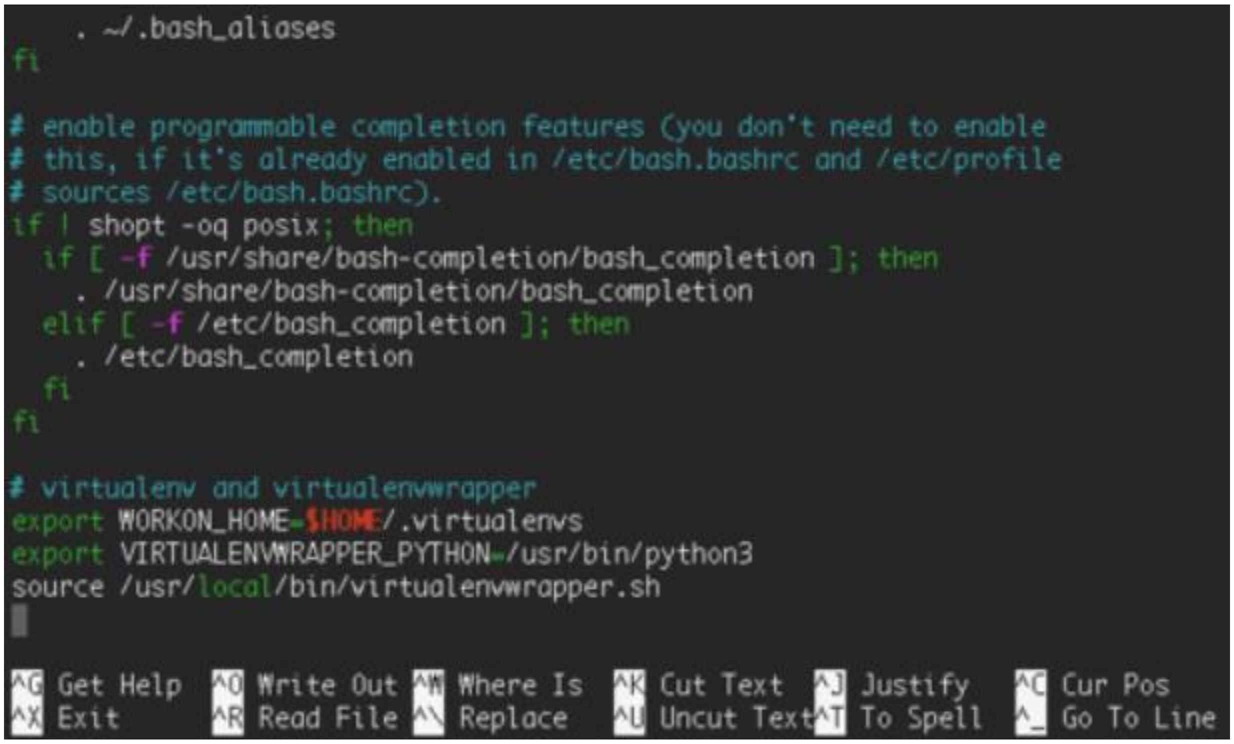
```
sudo pip install virtualenv virtualenvwrapper
```

Open your ~/.bashrc file once you have both virtualenv and a virtualenvwrapper installed:

```
nano ~/.bashrc
```

And attach at the bottom of the file the following lines:

```
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```



```
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

"Using the `nano` editor to update ~/.bashrc with `virtualenvwrapper` settings"

Press ctrl + x, y, enter

To implement changes in your current Bash session, reload your ~/.bashrc file from there:

```
source ~/.bashrc
```

Create the virtual environment of your Python 3 next:

```
mkvirtualenv cv -p python3
```

You need also install a PiCamera API if you have a Raspberry Pi camera module attached to your RPi.

```
pip install "picamera[array]"
```

Let's continue to download and unarchive the OpenCV source code for both the Opencv and opencv contrib repositories:

```
cd ~
```

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/4..1.1.26 zip
```

```
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.1.1.26.zip
```

```
unzip opencv.zip

unzip opencv_contrib.zip

mv opencv-4.1.1 opencv

mv opencv_contrib-4.1.1 opencv_contrib
```

You need to expand your SWAP space before starting the build. The SWAP increase allows you to build OpenCV using all 4 Raspberry Pi cores.

Go forward to open your /etc/dphys-swapfile:

```
sudo nano /etc/dphys-swapfile
```

Edit the variable CONF SWAPSIZE:

```
CONF_SWAPSIZE=2048
```

Initiate the swap service from there:

```
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
```

Ensure that the workon command is used in the cv virtual environment:

```
workon cv
```

Then go to the Python virtual environment and install NumPy(an OpenCV dependency):

```
pip install numpy
```

And customize your build from there:

```
cd ~/opencv
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
-D ENABLE_NEON=ON \
-D ENABLE_VFPV3=ON \
-D BUILD_TESTS=OFF \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D OPENCV_ENABLE_NONFREE=ON \
-D CMAKE_SHARED_LINKER_FLAGS=-latomic \
-D BUILD_EXAMPLES=OFF ..
```

Now that we are ready to assemble our OpenCV 4, the moment has come to start the compilation process with all four key components:

```
make -j4
```

Method for tensorflow is almost same. After Installing all the dependencies on Raspberry Pi Your Model can now be deployed on it. You simply have to run the files by Python.

CHAPTER 5

Database Development & Graphical User Interface (GUI)

5.1 Importance of a Database

The database development is one of the fundamental part for any grocery store as grocery stores need to keep a check on the inventory. The inventory is crucial for grocery stores to generate electronic receipts and keep track of how many products are left in the stores how many are to be ordered from the retailers. Moreover, an inventory will also help grocery stores in data analysis, the grocery stores can find out about the shopping trends followed by the customers and order products from the retailers according to those trends and demands.

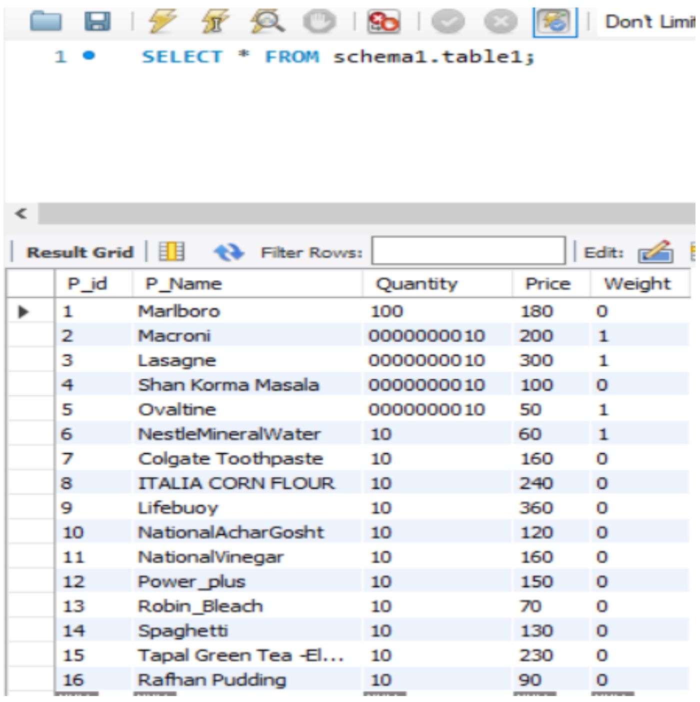
5.2 Database Design

Our database has 17 rows for 17 classes and 5 columns. The columns names are according to our needs which can be altered by store owner according to their needs.

The column names are as follows:

- P_iD,
- P_Name,
- Quantity,
- Price
- Weight

This database is developed on MySQL 8.0 community version. Although MySQL is available in two variants open source MySQL Community Server and proprietary MySQL Enterprise Server but we will be using MySQL community Server as it fulfils our needs which will be hosted locally.

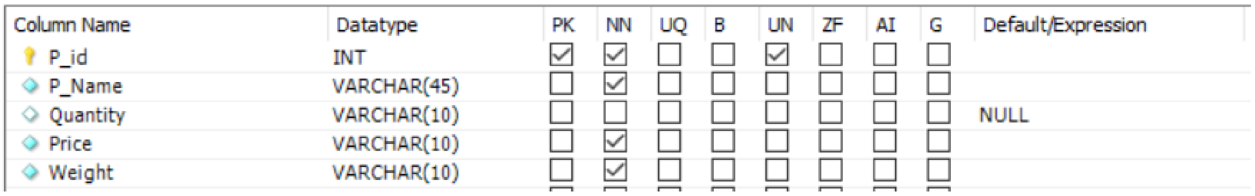


```
SELECT * FROM schema1.table1;
```

P_id	P_Name	Quantity	Price	Weight
1	Marlboro	100	180	0
2	Macroni	0000000010	200	1
3	Lasagne	0000000010	300	1
4	Shan Korma Masala	0000000010	100	0
5	Ovaltine	0000000010	50	1
6	NestleMineralWater	10	60	1
7	Colgate Toothpaste	10	160	0
8	ITALIA CORN FLOUR	10	240	0
9	Lifebuoy	10	360	0
10	NationalAcharGosht	10	120	0
11	NationalVinegar	10	160	0
12	Power_plus	10	150	0
13	Robin_Bleach	10	70	0
14	Spaghetti	10	130	0
15	Tapal Green Tea -El...	10	230	0
16	Rafhan Pudding	10	90	0

In the image below column details are shown:

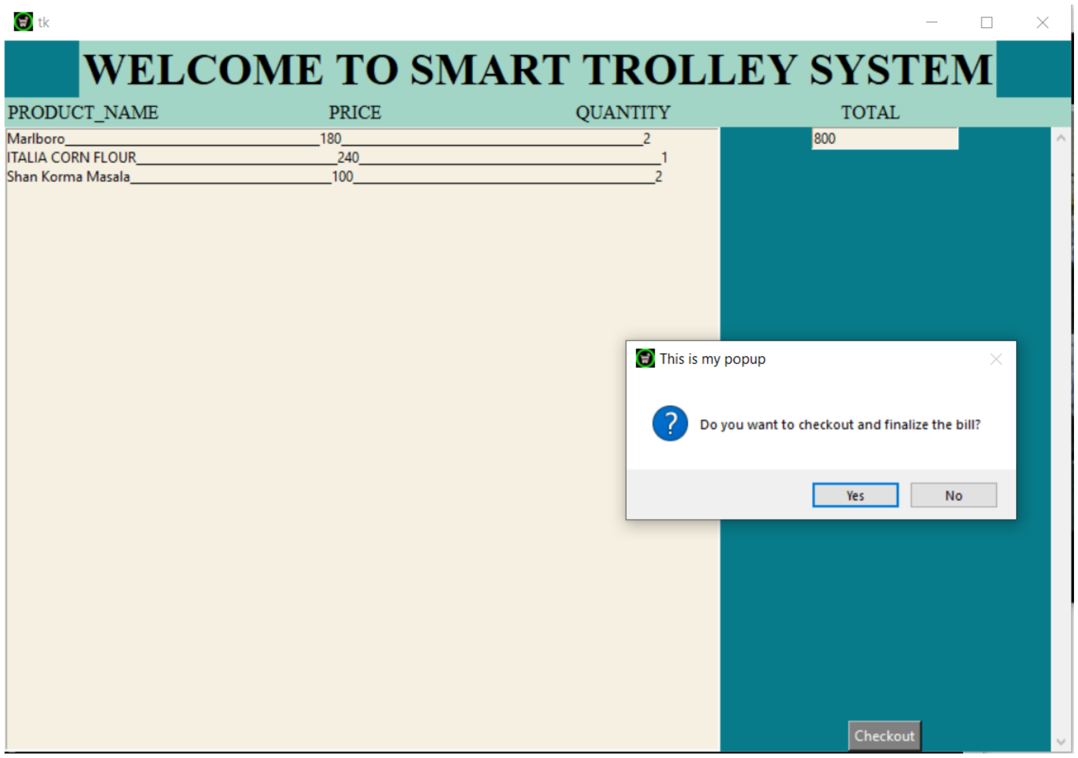
- The first column is that of the ProductID (P_id) is the primary key which means that it is unique for every entry (product) and its datatype is INT(integer) which is the default datatype for any primary key and it can never be NULL for any entry.
- The second column is that of the Product Name (P_Name). Its datatype is VARCHAR(45) which is the most suitable choice for entering the names of products and metadata is set to metadata NOT NULL
- The third column is that of Product Quantity (Quantity). Its datatype is VARCHAR(10) which is most suitable as per our requirement in this case and by default it remains NULL
- The fourth column is that of Product Price (Price). Its datatype is set to VARCHAR(10) which is the best possible choice for the purpose of entering the product prices as per the requirement in this case and it is set to metadata NOT NULL.
- Similarly, the fifth column is that of Product Weight(Weight). Its datatype is set to VARCHAR(10) which is most suitable as per our requirement for getting the weight of the products and is set to metadata NOT NULL.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
P_id	INT	✓	✓	☐	☐	✓	☐	☐	☐	
P_Name	VARCHAR(45)	☐	✓	☐	☐	☐	☐	☐	☐	
Quantity	VARCHAR(10)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Price	VARCHAR(10)	☐	✓	☐	☐	☐	☐	☐	☐	
Weight	VARCHAR(10)	☐	✓	☐	☐	☐	☐	☐	☐	

5.3 Implementation of Database:

We used this database in our project to generate a list of products placed in the trolley and their prices which is displayed on the screen placed on the trolley using Tkinter GUI. This is useful for the customer as the customer is aware of the products placed in the trolley and the total amount of money need to be paid.



```
tk
WELCOME TO SMART TROLLEY SYSTEM
PRODUCT_NAME            PRICE            QUANTITY            TOTAL
Marlboro________________180______________________2           800
ITALIA CORN FLOUR_________240______________________1
Shan Korma Masala_________100______________________2

This is my popup
Do you want to checkout and finalize the bill?
Yes   No

Checkout
```

Fig

5.4 Libraries Used

We have made our GUI using tkinter, Tkinter is a Tk GUI toolkit for Python. It is the default Python toolskit interface and is the de facto standard Python GUI. The usual Linux, Microsoft Windows and Mac OS X installations for Tkinter are provided in the program. Tkinter is a term derived from the interface Tk. Fredrik Lundh wrote Tkinter. Tkinter is freely available software under the license of Python. Tkinter is constructed around a complete Tcl interpreter, which has been integrated in a Python interpreter as a Python wrapper like with most current TK bindings. Calls from Tkinter are converted into Tcl instructions are feed to the embedded interpreter, enabling a single application to blend Python and Tcl. A range of popular GUI library options, including wxPython, PyQt, PySide, Pygame, Pyglet, and PyGTK, are available.

So we started by installing the necessary libraries, the libraries are as follows,

```
import time
import tkinter as tk
from tkinter import *
from threading import Thread
import mysql.connector
from tkinter import messagebox
```

We used time library for putting up a necessary delay between the scanned item. Then for importing tkinter and its functionalities the next two libraries are being installed. For the connection with the database we have used **mysql.connector**. For the use of threads, we have imported it and finally a messagebox library is used when the checkout button is pressed.

5.5 Threading for Parallel Processing

While making GUI, first we created an instance of thread, a thread resembles the previously mentioned sequential programs. There is also a start, sequence and end for a single string. There is a single point of execution at any moment during the course of the thread. But a thread itself is not a program; it cannot execute a thread alone. It's running in a program instead. Since we have three programs running parallel, one python file for making the bounding boxes and predicting the labels for the concerned product other python file for the detection video and the third one for the GUI, so for the smooth running of programs during the real time we have created the threads.

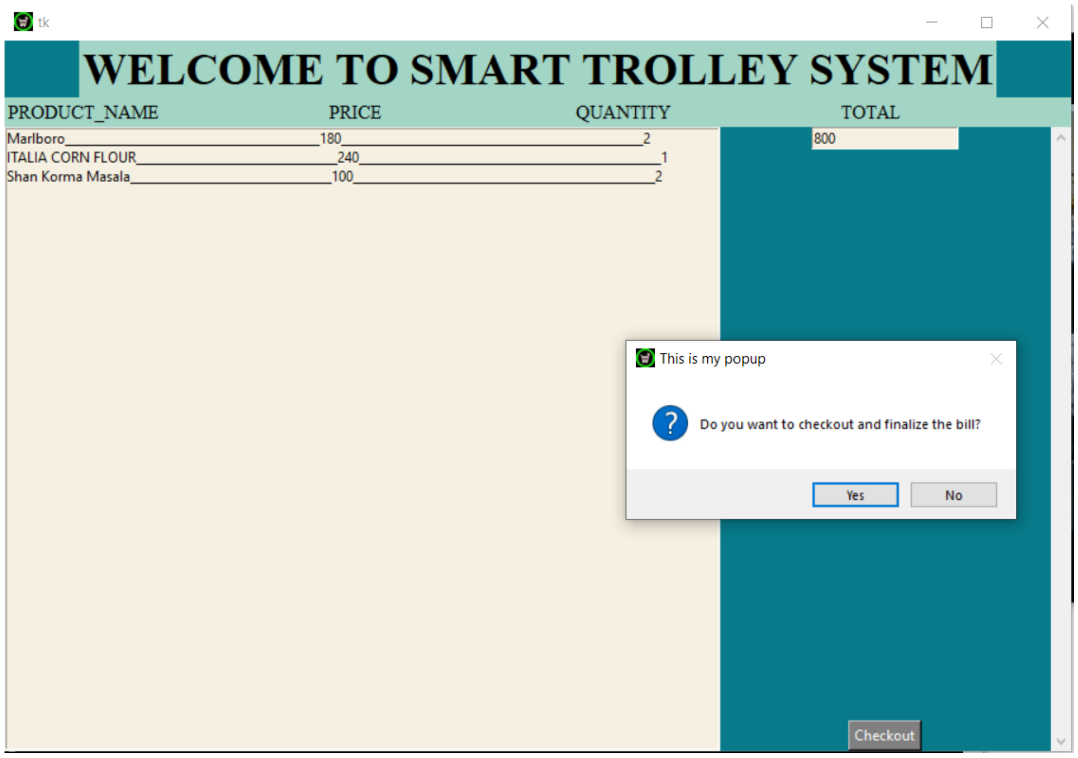
5.6 GUI Design and Queries Used

Once the thread is instantiated, our next step is to make a basic structure for our GUI. For that we have used list box, checkbox, labels and entry widgets etc. The GUI looks like this after adding the basic structure. We have defined list box with side=” LEFT” given the width to be 100% and defined the labels for product name, price and quantity. On the top we have our title defined.

Alongside this we have a total box which shows the real time reading of the total money to be paid with respect to the price and quantity of the products being listed in the inventory, shown in the listbox. A scrollbar is displayed on the right hand side and is connected with the listbox so if the items in the listbox increases we can easily view them just by sliding the scroll bar.

A checkout button is made at the end to finalize the bill. On pressing this button, a response box is displayed to confirm that whether a person really wants to end the shopping and

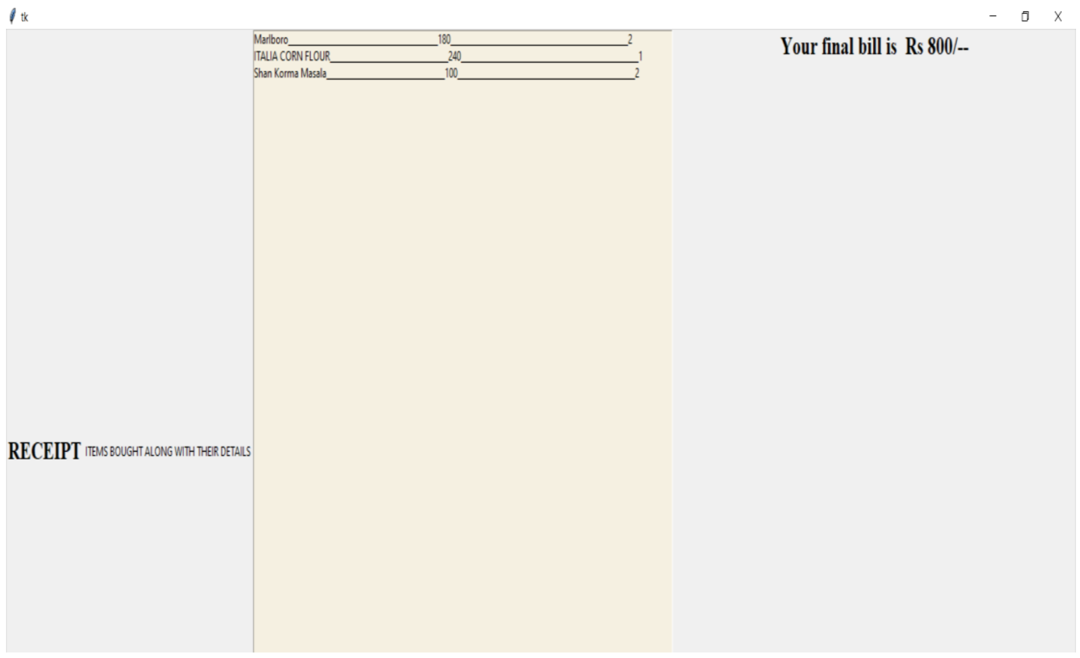
should be displayed with the final bill or he wants to continue with the shopping. If the person confirms the end of shopping, a bill is generated in the separate window, with the details of the item bought and the final bill to be paid else a message is displayed, “keep on shopping”.



```
Marlboro___________________________180_____________________________________2
ITALIA CORN FLOUR__________________240_____________________________________1
Shan Korma Masala__________________100_____________________________________2
```

800

Do you want to checkout and finalize the bill?



RECEIPT ITEMS BOUGHT ALONG WITH THEIR DETAILS

```
Marlboro______________________________180____________________________________2
ITALIA CORN FLOUR________________________240____________________________________1
Shan Korma Masala________________________100____________________________________2
```

Your final bill is Rs 800/--

We have added a time delay of about 3 seconds so that the same product does not get scanned multiple times and to avoid random entries. For two instances we have made a connection with the database. First time when we have to fetch attributes from the database and to list them in the inventory and for the second time when we have to update our database after a customer checkout.

The connection with the database is done by specifying host, user, password and database. Once the item gets identified call is made to database for the attributes of the respective product. The attributes contain the product name, price and the quantity depends on the no of times the object is being scanned. We have put a check that if the item already exists in the inventory then a separate instance should not be created but a +1 is done in the quantity.

For the updation part that was done using sql queries. First we have selected all using SELECT * query from the database. Then I have picked up the quantities of the product in our inventory and subtracted it with the total quantity of the product in our database. After that we updated the database using UPDATE query and after updation I have listed the items again bit now with the quantities being updated.

Chapter 6

Business Plan

6.1 Cost Effectivity

- A regular shopping cart costs between 10,000 PKR - 20,000PKR depending on its size.
- Our proposed shopping cart will cost around 30,0000PKR – 40,000 PKR including the smart system.
- Despite it being more expensive than a regular shopping cart it will result in no employee requirement for cash counters which in turn will lead to the whole product being cost effective as compared to a regular shopping cart.
- Furthermore, when all the manufacturing components will be bought in bulk for manufacturing a huge number of smart systems for trolley it will reduce the overall price of the shopping cart.

6.2 Market Size

International Market:

- There are roughly **2 million** supermarkets in the world and this number is increasing significantly

with the increasing demand and the modernization.

Local Market:

- There are **Thousands** of supermarkets in Pakistan
- Pakistan's retail industry is growing substantially as major brands and outlets join the market.

In addition to dealing with their largest international rivals, local merchants are expanding their operations.

6.3 Market Adoption Strategy

Our formal market strategy would be based on the following key points:

- Form partnership with local supermarkets
- Form partnership with high end brands to target larger audience.
- Form partnerships with the vendor of supermarket equipment.
- Social media advertisement for target audience.

6.4 SWOT Analysis

Strengths:

- Cost effective.
- Large-scale evaluation approach.
- Market Research.
- Privacy.

Weaknesses:

- No human interaction.
- Swappable battery.

Opportunities:

- No local competitor.
- Existing solutions are cost-prohibitive.
- Market needs this service.

Threats:

- Fraud(stealing).

Future Work

- Adding weight sensor to overcome fraud-For the purpose of security and check of fraud while shopping by any of the customers. We will be implementing the weight sensor in our smart trolley system which would enable us to calculate the weight of the objects that have been entered in the smart shopping trolley so that we can compare it with the total weight of the products from the database that the user has put into the smart cart.
- Adding Payment Gateway: **payment gateway** is the technology that captures and transfers payment data from the customer to the acquirer and then transfers the payment acceptance or decline back to the customer. It acts as an interface between a merchant's website and its acquirer. The Customer Can Scale quickly with our efficient online payments solutions & let your customers choose from options such as M-Wallets, Credit/Debit cards & OTC and let them pick the way that works best for them.
- Deploying this model on Jetson Nano-Since Jetson Nano has better Computational and processing power. Since the model that is being deployed in our prototype requires high computational power therefore we would be implementing the model that we have trained on Jetson Nano in future for better performance and running of our model. Since Jetson Nano is greater and much more efficient in case of its performance and deployment of such a graphical model it would be better than Raspberry Pi that we are using.
- Transfer learning- Transfer learning is a machine learning research issue that aims to store and apply information obtained in the course of addressing one challenge. We have made a crop flag that basically crops the detected object from the video feed and save it in the folder. Which can then be used for transfer learning.

Conclusions:

The precision of Amazon Go, known as the representative of unmanned businesses, is considered to be successful, and Amazon Go's ideas and talents have been acknowledged by the public, but spreading extensively is expensive. There is also a restriction on the number of clients who can enter at the same time. The smart cart system suggested in this study outperforms existing unmanned store options in terms of cost-performance ratio. Even if the capacity increases, the proposed system is unaffected. It also consumes a little amount of CPU power. Furthermore, unlike the traditional way, RFID does not have to be attached to all items. Speed and accuracy are trade-offs in the world of product detection, however because this system required real-time processing, it was done using YOLO. It is expected that in the future, this section can be substantially enhanced by updating and updating the object detection model.

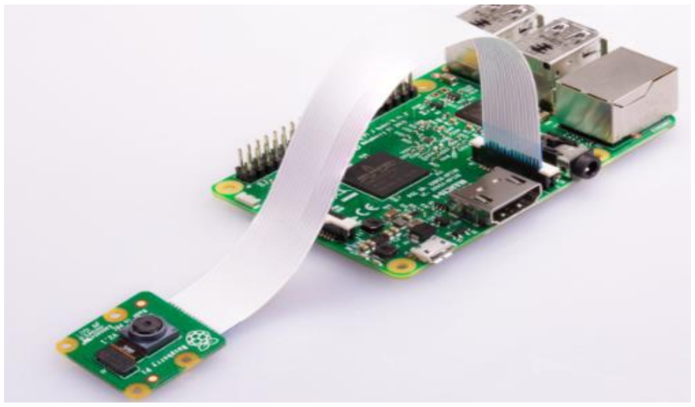
Appendix:

Raspberry Pi v4:

Rpi Camera V2:

Jetson Nano 2GB:

Rpi Camera connected to pi 4:



References

[1] Wankhede et al, Wukkadada, and Nadar. Wankhede K, Wukkadada B, Nadar V. (2018). Just walk-out technology and its challenges: a case of amazon go. In: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE, pp 254–257.

[2] Chiang et al, You, Lin, Shih, Liao, Lee, and Chen. Chiang HH, You WT, Lin SH, Shih WC, Liao YT, Lee JS, Chen YL (2016). Development of smart shopping carts with customer-oriented service. In: 2016 International Conference on System Science and Engineering (ICSSE), IEEE, pp 1–2.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars. (2008). Computer Vision and Image Understanding. Volume 110, Issue 3, pp 346–359. https://doi.org/10.1016/j.cviu.2007.09.014.

[4] D. Fleet et al. (Eds.) (2014). Recognizing Products: A Per-exemplar Multi-Label Image Classification Approach, pp. 440–455. http://www.vs.inf.ethz.ch/publ/papers/mageorge_products_eccv2014.pdf.

[5] https://ieeexplore.ieee.org/document/7561176

[6] Annalisa Franco, Davide Maltoni, Serena Papi. (2017). Expert Systems with Applications: An International Journal. Volume 81. Issue C. pp 163–176. https://dl.acm.org/doi/10.1016/j.eswa.2017.02.050

[7] https://ieeexplore.ieee.org/document/1640452

[8] Manisha Agrawal , Nathi Ram Chauhan. (2017). 3D Object Recognition for Automated billing in a Supermarket using Hybrid PCA-SIFTFREAK Algorithm. https://www.ijareeie.com/upload/2017/july/75_E60707654.pdf.

[9] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. (2004). YOLOv4: Optimal Speed and Accuracy of Object Detection.

[10] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh (2020). CSPNet: A new backbone that can enhance learning capability of cnn. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop).

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition.

[12] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. 2018. Path aggregation network for instance segmentation, pp 8759–8768.

[13] Joseph Redmon and Ali Farhadi. (2018). YOLOv3: An incremental improvement.

[14] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, ´ Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection, pages 2117–2125.

[15] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. (2004). YOLOv4: Optimal Speed and Accuracy of Object Detection.