Autonomous Checkout System

(Final Year Project)

Supervisors

Dr. Junaid Akhtar (Namal)
Dr. Muhammad Hussain (KSU)

Group Members

Muhammad Hamza
Mubashaar Nisar

Namal Institute Mianwali, Pakistan
Department of Computer Science

Contents

Abstract

A system is proposed for the retailers with a computer vision and machine learning based system that supports the records of buyers and items they pick up in order that the customers may be charged perfectly and save the retailers from shoplifting. The system will detect and arrange the products customers picked and keep records of the customers and finally create a bill for each customer once they reach the checkout point. The dataset used to train a Deep Convolutional Neural Network is curated using their different kinds of images from real environment videos. CCTV video with the field of view of 45 degrees (Parallel to the products rack), Street view imagery (Perpendicular to the products rack) and CCTV captured videos with the field of view of 45 degrees (Diagonal to the products rack). One solution this solution on the local system showing 99.9 percent accuracy for human detection and tracking, 89 percent accuracy for picked or unpicked products. Publicly available packages for deep learning like TensorFlow, Keras, OpenCV, Django and Django Rest Framework, and ReactJS are used to construct an Autonomous Checkout System from a CCTV video. Pertinent to mention is the lack of expensive sensing de- vices (CCTV camera, real products, Products rack etc.) in our implementation, a stark shift from many of the previous studies along with demonstrable detection results with excellent potential for scalability of the proposed solution to work with hyper-spectral sensory data.

Acknowledgement

In the start, we are very thankful to Allah Almighty that he helped a lot to keep us consistent and persist in our goal achievement. I would like to say thanks to all those who encourages and enables us complete this report. By accompany of honest, loyal and hardworking group, makes this possible. We would like to express our special gratitude to our final year project supervisors: Dr Junaid Akhtar and Dr Muhammad Hussain. I have to cherish the consel given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices. Without their contributions, we would never be here as they assist a lot in this project specifically writing the report. I would like to indicate my grateful to my parents or their whole heartedly cooperation and encouragement that help me in the finalization of this project. I would like to express my deep hearted thank to industry persons for giving me such awareness and time. My thanks and appreciations also are for my colleague in developing the project and people who have voluntrily assisted me out with their abilities.

Problem Statement

According to reports, there are a total of 5,327 utility stores in the country out of which 4,470 are incurring losses. In Islamabad 494, Abbottabad 428, Sargodha 607, Sukkur 280, Quetta 265, Karachi 364, Lahore 508, Peshawar 790, and Multan has 679 of these stores which are facing losses. These are the losses that were facing every utility store in Pakistan. Also, everyone knows the mismanaged queues in Pakistan in every store. This project envisions the possibility to walk in, grab whatever you like and leave with a self-checkout system. But for starters, we want to confine the project to shops and bakeries that have a few products like Lays chips on racks and glass cased cold drink freezers located outside the shops. Given Pakistani society's ethical standing and economic situation of a common shopkeeper, shoplifting is a legitimate and serious issue. Some retailers do install cameras to hold a check on their products however that's not enough because there needs to be someone who's manually looking at the display screen all the time. Keeping in view these serious issues there is a need to develop an intelligent camera-based system that keeps track of customers and the items they picked from outside the store, to help both the retailers as well as shoppers.

Objectives

The main objective of our project is to provide retailers with a computer-vision and machine learninf based autonomous system that keeps track of buyers and items they pick up in order that the customers may be charged rightly and save the retailers from shoplifting. First of all, the system will detect the human from CCTV video and also keep track of the customer in front of the camera. If the customer interacted with the products rack then the system would detect and classify the products customer picked and keep track of the customers and products picked up by the customer and finally generate a bill for each customer once they reached the checkout point.

Introduction

The autonomous checkout system is computer vision and machine learning based our models for person detection, tracking person and products, item detection, item classification, and ownership resolution, all working together to visualize which person has picked what item in real-time. The system deals with the detection of customer with the help of CCTV cameras to capture all actions in the form of images and videos so products and human's detection is possible to reduce the shoplifting of retailer. The customer picks the product of his choice that system allows customer to buy product in the absence of shopkeeper. Products are classified. He pays final payment. Final payment is received by the admin. The system generated auto receipt to customer. This system also permits retailer to keep an eye on all procedure by viewing on online dashboard that what is happening inside all the business. The methodologies that are used including Computer Vision, Neural Networks, Data Augmentation, Machine Learning, Artificial Neural Networks, CNN (Convolutional Neural Networks), Convolution layers, Pooling Layers and Fully (Flattening) Connected Layer. We have used technologies including TensorFlow, Keras, OpenCV, Google Cloud Platform, Django, Django REST framework and React JS. We have used tools including HTML5 with Jinja Templates, CSS4, Java Script, jQuery, Python with Django Framework, Django Rest Framework for API's, PostgreSQL and PGAdmin-4 backend server for Database.

Methodologies

Computer Vision

Computer vision, in its simplest form, is the ability of a computer to "see" and process visual data. More formally, it is an interdisciplinary field of study aimed at understanding why the human visual system works and ultimately replicating it. It is the process of extracting, analyzing and understanding visual information present in images or videos. A digital image can be defined as a continuous sample of light radiations reflected from different objects onto a rectangular array of pixels. Digital signal processing can be applied to extract information and generate meaning out of these numbers.

With the deep learning revolution, the domain of computer vision has pro- gressed exponentially and can now boast about object detection and scene com- prehension from videos and images, alike. In order to understand an image, three primary levels of abstraction are utilized

to make sense of the sensory data, at the bottom includes the basic components of an image i.e. edges, tex- ture elements or regions, a higher level of abstraction is inclusive of boundaries, surfaces and volumes, while the highest level of comprehension includes objects, scenes, or events recognition.

Neural Networks

A neural network mimics the human brain and learn just like humans. The neural networks learn through an algorithm by collecting or providing a new data. A neuron also is known as a node and neuron is the basic part in a neural network. These neurons combined to build a large mesh network. The node receive input from an external source and generate an output or also from some other node and generate an output. Every node has some unique weight (w), which is assigned based on its relative importance in the whole network to other input. And finally apply a function to the weighted sum and receive final out from the network.

Methodologies Data Augmentation

Any neural network is a combination of "neurons" which is organized in layers and those layers are defined as follows:

Input Layer

It takes predictors/inputs and attaches coefficients to them which are called "Weights".

Hidden Layer

It makes the neural network non-linear and improves performance. Neural-Network can be linear with no hidden layer.

Output Layer

It outputs the results generated from weighted predictors passed through hidden layers, if any.

Back-propagation

A procedure is performed to adjust the neural network weights so as to minimize the different or loss between actual output and desired output of neural networks.

Error

It is calculated by taking the difference between desired output and output predicted by the neural network. It creates a gradient descent which can help in altering weights.

Weights

These are altered by method back-propagation. In this way, an error can be removed. Then gives multiple iterations to this process and finally an accurate model is trained. There are several types of neural networks, each having specific use and flexibility. The most common neural networks are; feed-forward neural network, recurrent neural network, and the convolutional neural network and Boltzmann ma- chine network.

Difference between Traditional ML and ANN

The major difference between traditional ML techniques and ANN is feature extraction'. In traditional ML techniques, features are extracted manually by using different methods but in ANN features are extracted automatically by adjusting the weights. Each layer in ANN learns particular features and uses it to identify objects.

A Rectified linear unit (ReLU) takes a real-valued input and threshold it with zero, replacing every negative value with zero. This operation has been used after every convolution operation. As convolution is a linear operation but most of the real-world problems are non-linear, so for non-linearity, ReLU is introduced. For a processing task in deep learning, neural networks cannot be programmed directly like other algorithms. Neural networks use some strategies to learn information. For example: supervised learning, unsupervised learning and reinforced learning.

The sigmoid function generates similar results to step function in that the output is some between 0 and 1. The curve crosses 0.5 at z = 0, which we can set up rules for the activation function, such as: if the sigmoid neurons output is larger than or is equal to 0.5 then its output is 1, if the output is smaller than 0.5, its output is 0. Neural Networks' behavior is

determined by its neurons' particular transfer functions, by learning rule and architecture. Transfer function, also called Activation functions, defines the output of given input or set of inputs to a node of neurons is shown in Figure below:

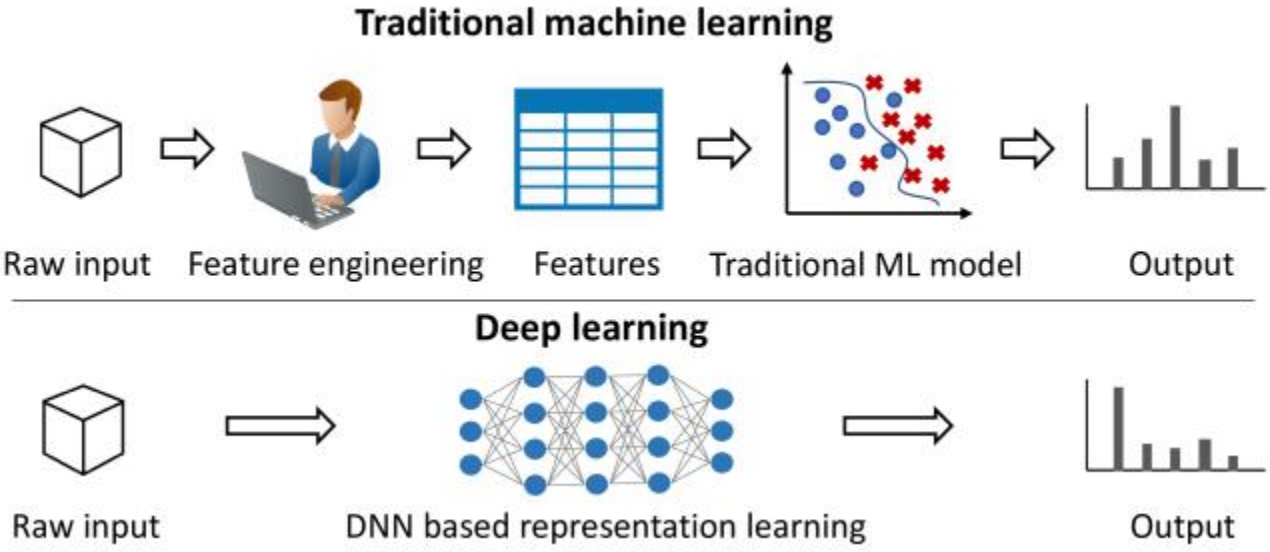


Figure 1: Traditional ML vs ANN

Neural Networks' behavior is determined by its neurons' particular transfer functions, by learning rule and architecture. Transfer function, also called Activation functions, defines the output of given input or set of inputs to a node of neurons is shown in Figure below:

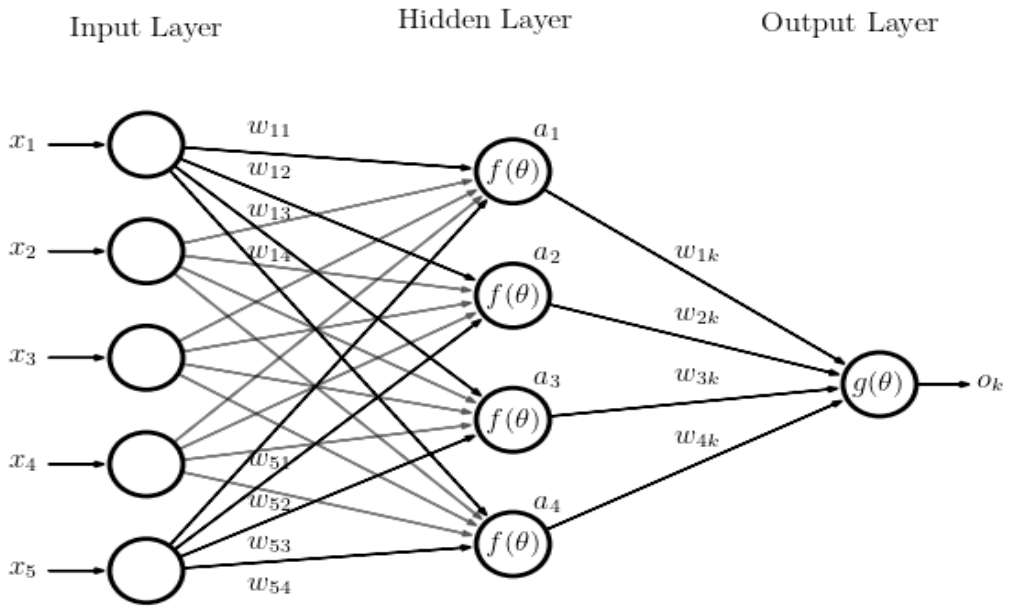


Figure 2: Basic Function of Neural Network

Applications of the neural networks can be summarized into classification or pattern recognition, prediction and modelling.

Convolutional Neural Networks (CNN/ConvNets)

Convolutional neural networks are the same as other neural networks, made up of neurons and synapses (weights). Each neuron receives an Input, multiplies it with weights and applies an activation function on it. Error is calculated and weights are altered according to the desired output. The change between CNN and other neural networks is ConvNet architecture. ConvNet is assumed to take inputs in the form of images. CNN/ConvNet architecture has three main layers; convolutional layer, pooling layer and fully-connected layer.

Convolutional layer

It is a mathematical operator which multiplies the image matrix with ker- nels/filters to extract features from an input image. Figure below shows the multiplication of the image pixels matrix and filter matrix.

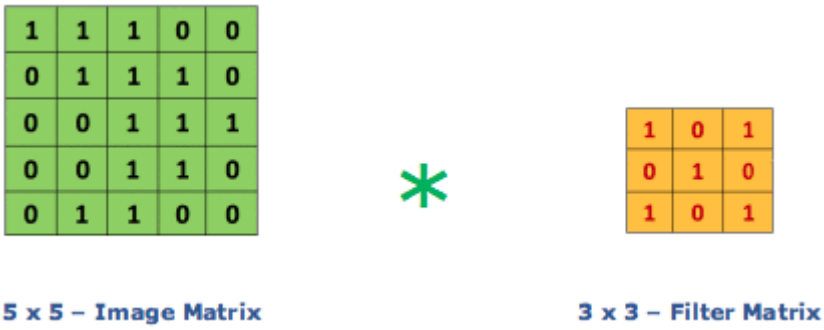


Figure 3: Convolutional Layer Operation

Filter matrix move across the whole matrix of image and gives convolved feature matrix as shown in Figure 1.3

At this layer, Rectified Linear Unit (ReLU) such as f(x)=max (0, x) is applied element wise to introduce non-linearity.

Pooling Layer

In a case of large or complex images, pooling layer reduces the number of features without losing important features information. It has three types; Max pooling, Average pooling and Sum pooling.

Fully Connected Layer

Fully connected layer combines all the features coming from convolutional layer and pooling layer and develops a trained model. Then the activation functions like sigmoid or softmax are applied to detect and classify objects.

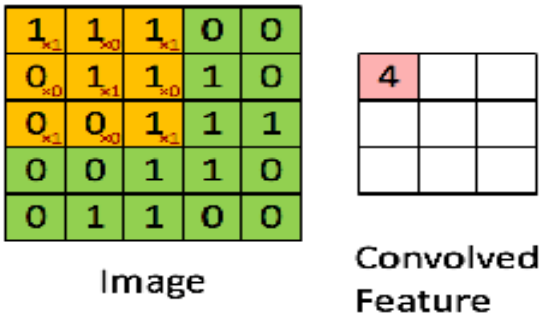


Figure 4: Convolved Features

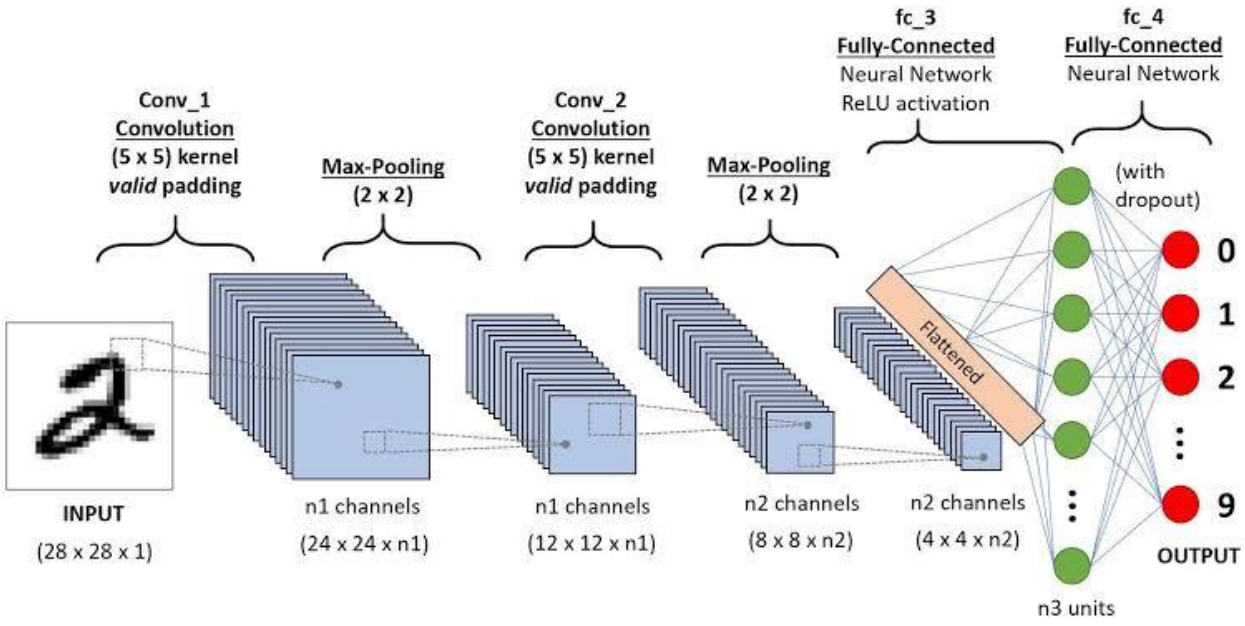


Figure 5: The Whole Process of Convolutional Neural Networks

Literature Review

Machine Learning

Since the beginning of the industrial revolution, machines have been the helping hand of humans. We use various machines from start to the end of the day. Almost two decades before, machines were dumb and operating manually but now they are more flexible and intelligent. Understanding the texts, driving cars, identifying specific persons etc. all are the tough tasks which machines are doing now. This is all happening because of 'Machine Learning'. According to Paper Name machine learning is a branch of artificial intelligence which makes machines learn and act like humans and improve their learning in an autonomous way by feeding them data in the form of observations, statistics and real-world interaction. It has been categorized into the following groups.

Supervised Learning

In supervised learning, the machine concentrates on learning patterns by connecting the relationship between variables and known outputs. It works by feeding the machine with input features(X) and the correct value of output(Y). Then the machine learning algorithm makes patterns and develops a model which will give an intelligent output on new data. A very simple example is a training system on images and labels. Then in future, if you'll give a new image, it will intelligently recognize the new image.

Unsupervised Learning

In unsupervised learning, we do not feed machines with all the variables and known outputs. Instead, we let the machine uncover hidden patterns and generate labels through unsupervised learning algorithms. One of the examples is the K-means clustering algorithm which groups the data points which have similar features. Figure-2 below shows the difference between supervised and unsupervised learning.

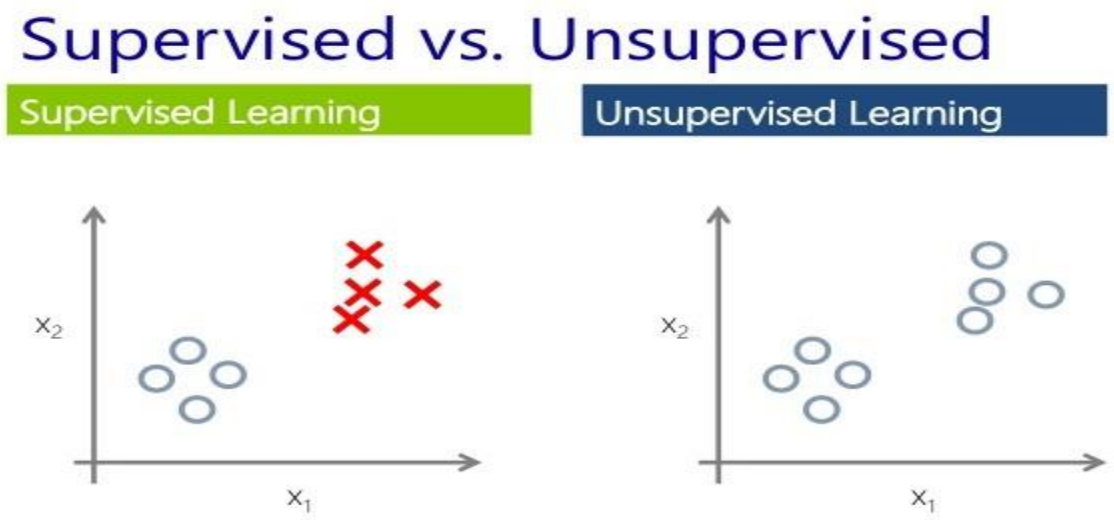


Figure 6: Supervised and Unsupervised Learning

Reinforcement Learning

This type of learning is quite different and advanced as compared to supervised and unsupervised learning. In reinforcement learning, machines continuously improve their model by leveraging feedback from the previous iteration. It can be explained with an example of a self-driving car. If in any case, the car avoids a crash then it will learn the value of this action. Next time, if the car will face the same situation then it will take the same action to avoid that accident.

Machine Learning Contributions

Machine learning plays a significant role in our daily lives. From transportation to health care, education to public safety, it has improved our lives. In transportation, autonomous cars are in the market. Cars are becoming better drivers than men and saving a lot of time for other activities. In Healthcare, we have monitoring devices and mobile applications which keep the track of our daily health routine. Moreover, robots are also assisting doctors in various surgeries. In Education, we have interactive tutoring machines capable of teaching science, math and other disciplines. Further, classrooms have an in- telligent system which continuously informs the teacher about the interest rate of students on the basis of facial expressions. In Public Safety, smart cameras and drones are being used for surveillance. Moreover, algorithms for detecting and avoiding financial

fraud and privacy interference are also available. Besides these, machine learning is contributing in other areas also. One of them is in advertising.

Proposed Framework

The proposed system includes person detection, picked and unpicked classification, product classification, and Hand Localization, these all models will work together to visualize which person has what product in real-time.

1. Person Detection
2. Product Picked Unpicked classification
3. Product Classification
4. Hand Localization
5. Desktop Application for end User

Work being done

The work that we have done so far is real-time person detection, product picked and unpicked classification, product classification and hand localization. For the purpose of person detection, we have used the MobileNet-SSD pre-trained model, and for our second part picked and unpicked classification we have used CNN Binary Classifier trained on our own dataset that we have described ahead. Product calcification done using CNN model trained on custom dataset.

Picked-Unpicked Classifier

Dataset Collection

We have collected datasets of two different environments, one using DSLR and the other using mobile-cam. In the first environment, we had lays chips rack that was filled with chips packets, for this environment we have shot two videos. In the first video, only one person goes and picks up a packet of chips, while in the second video, more than one person used to pick up chips packets. In the second environment, we had a shelf in which two different kinds of products were placed. For this environment too, we have shot two videos, In

Figure 7: Dataset Collection-1 Frames

the first video, only one person goes and picks up a product, while in the second video, more than one person used to pick up products.

Figure 8: Dataset Collection-2 Frames

Dataset Labeling

Image labelling or annotation is the process of defining regions in an image and giving them a textual label for creating ground truth. One of the many magnificent features of the humans' visualization is to create a rich description of a scene within a glance at an image. They can tell what are the objects and their associated attributes in the image. For example, an image can be defined as "containing a person, picking a product from the rack". Moreover, humans can effortlessly describe each object in the image. During the past few decades, one of the fundamental objectives of computer vision research has been to replicate this ability, resulting in in-depth

studies of computer vision problems including detection of objects in a scene by describing them using their attributes and creating the ground truth.

OpenCV is the most common library for Computer Vision problems. We have extracted 2400 frames from both videos using OpenCV. After extracting frames from the videos we labelled these frames manually in bases of product picked or unpicked. Our dataset split for training is 80 percent and 20 percent for testing.

The Streamlit

Streamlit is an open source application framework that is particularly designed for Machine Learning engineers who are working with Python. It allows you to create a remarkable looking application with only a few lines of code . It supports reloading so your application updates live as you add and save your file. It means that there is no need to mess with HTTP requests, HTML, JavaScript, etc. All you need is your favorite editor and a browser.

We used Streamlit for a live demo. Using Streamlite we show the sidebar which represents the all testing images using a simple slider. In the center vertical order we show the live demo of videos: first single person in frame, two person in one frame, and Single person in different environment dataset video. After videos we show the complete procedure on a single frame. Before processing a single frame, Person detection from a single frame, person cropping, and then show the result picked or unpicked.

Flow Chart

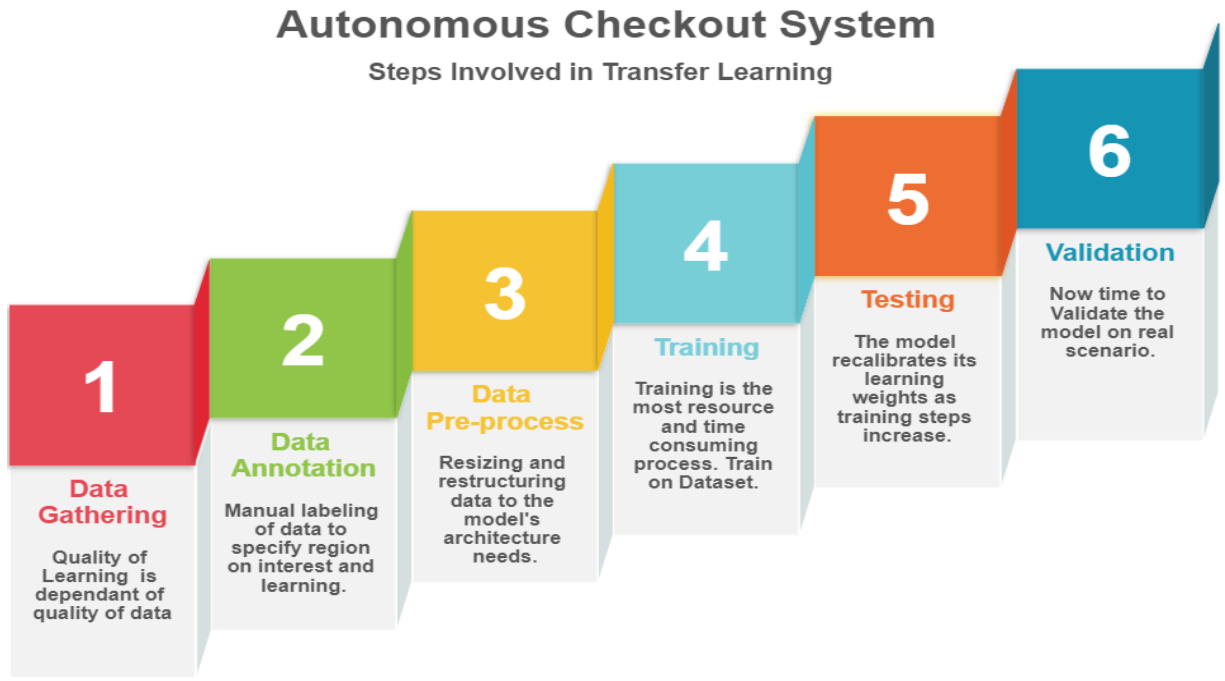


Figure 9: Steps we are Following

Picked and unpicked classification using Keras

Our essential undertaking is to make an algorithm to classify whether a picture contains a person with a picked product or unpicked. The input for this task is pictures of picked or unpicked from the training dataset, while the output is the classification accuracy on the test dataset. The given dataset for this task is gathered by ourselves, clarified previously. Our training set contains 2,400 pictures, including 1,200 pictures of picked and 1,200 pictures of unpicked, while the test dataset contains 400 pictures.

Our principle task is to learn a classification model to determine the decision boundary for the training dataset. The input for the learning task is pictures from the training dataset, while the yield is the learned classification model. Our end task is to apply the learned model to classify pictures from the test dataset weather picked or unpicked, and then evaluate the classification accuracy.

Deep Neural Networks

The CNN is a sort of deep architecture that has accomplished extraordinary performance in tasks like document recognition and image recognition. Not the same as customary BP Neural Networks, which contains input layer, hidden layers, and output layer, CNN likewise contains Convolutional layers and Max Pooling layers.

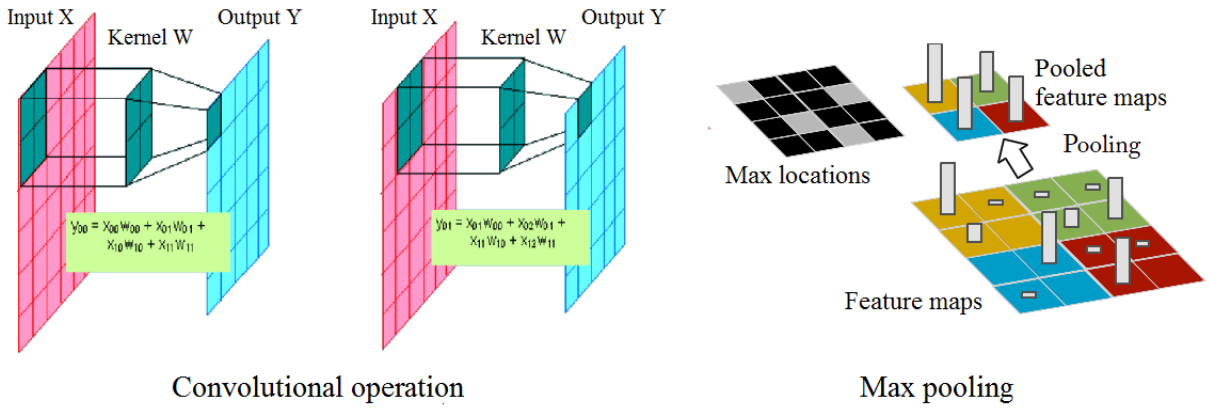


Figure 10: Convolutional Operation Max Pooling

Convolutional layers contain many feature maps which are based on two dimensional hidden nodes. Each feature map claims a weight matrix called kernel, and different feature maps owns different kernels. Kernels do convolutional activity with each feature map in previous layer (layer j), at that point we summarize them and put it into sigmoid function. The output is of the pixel value in layer that is j + 1. With various kernels, we can learn various representations of information and the measure of parameters isn't expanded exponentially with the number of hidden nodes and layers.

When a feature has been detected, its exact location becomes less significant. Just its approximate position comparative with different features is relevant. Not exclusively is the exact position of every one of those features unimportant for recognizing the pattern, yet it is additionally conceivably unsafe in light of the fact that the positions are probably going to change for various instances of the pattern. Max pooling layers do sub-sampling procedure on feature maps. For each four pixels, we just hold the maximum value, with the goal that the size of feature maps will be half of the first size. Sub-sampling lessens the resolution of feature maps and decreases the affectability of the output to movements and distortions with the goal that the

model will be increasingly robust. Max pooling activity can be joined into convolutional layers, and we needn't bother with extra layers to do sub-sampling.

Model

The original model contains 4 layers and the last two layers are one fully con- nected layer and output layer. We added one more fully connected layer and Dropout layer to avoid over fitting. This deep convolutional neural network trained by ourselves on our own dataset.

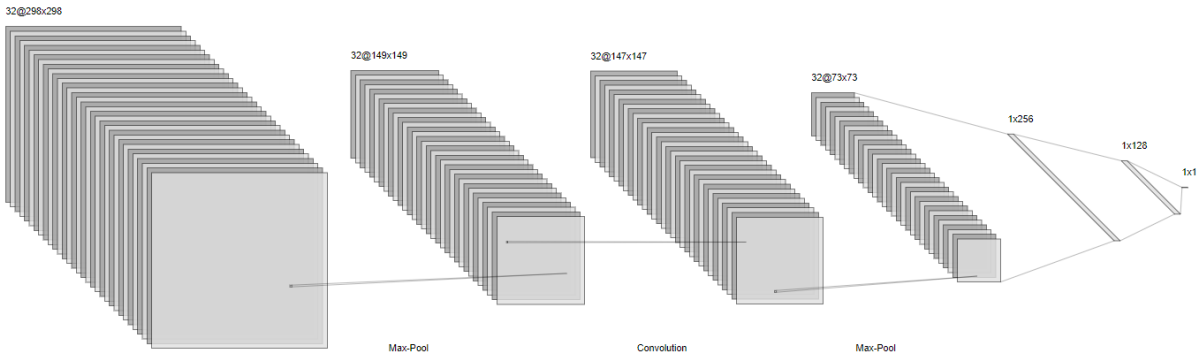


Figure 11: Model View

The architecture of this model is outlined in the above figure. In the first place, input pictures are normalized into 298 x 298 then the first convolutional layer filters the 298 x 298 x 3 input picture with 32 kernels of size 3 x 3 with a stride of 4 pixels. The second convolutional layer takes as input the pooled output of the first convolutional layer and filters it with 32 kernels of size 3 x 3. The Dense layers have one layer with 256 neurons and second layer with 128 neurons and finally the output layer. In the output layer we have 1 neuron and sigmoid activation, which means this will give us out as 0 or 1 (binary ouput).

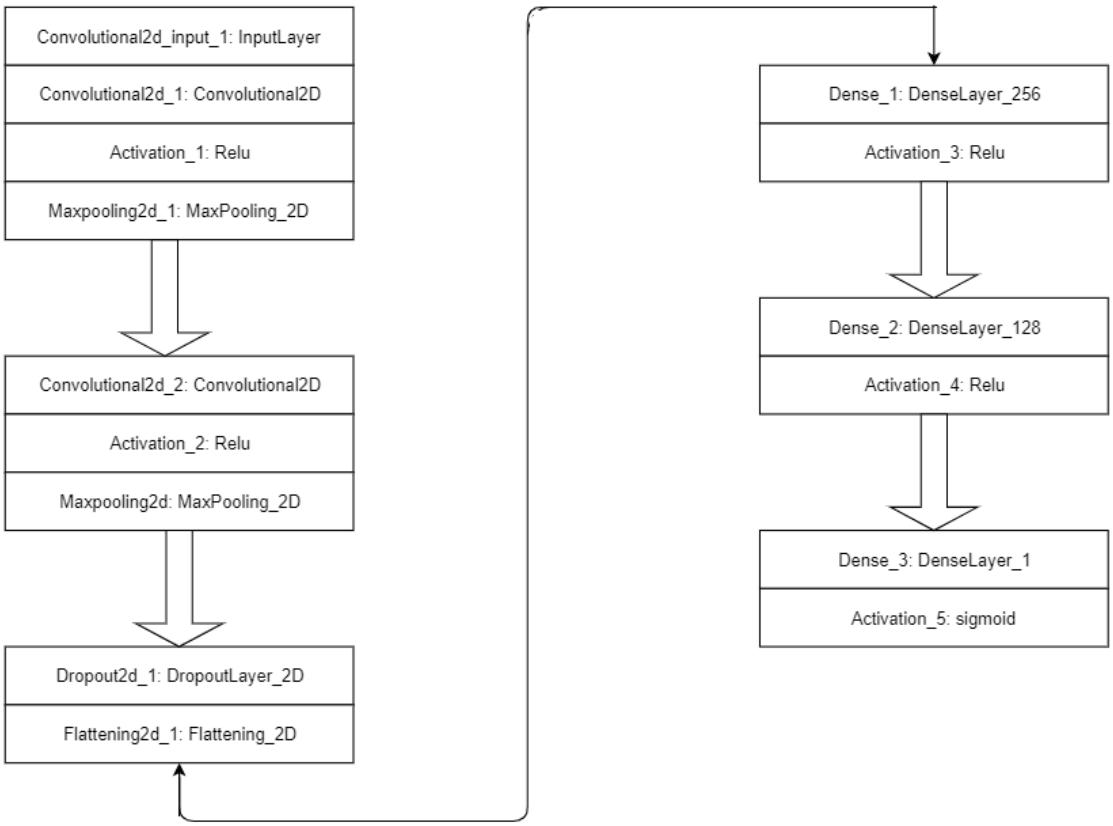


Figure 12: Picked-Unpicked Model Architecture

Model Architecture

Convolutional Layers

- 2 Layers
  - 32 Filters of 3x3 size on
    First Layer
  - 32 Filters of 3x3 size on Second Layer All convolutional layers have pooling of 2. Dropout Layer that keeps up probability of 0.5.

Fully Connected Layers

- 3 Layers
  1. 256 Neurons with Relu Activation Function at first layer
  2. 128 Neurons with Relu Activation Function at second layer
  3. Neuron with Sigmoid Activation Function at output layer

Results and Analysis

By using this model, we have got very impressive results with the training accu- racy 98 percent and testing accuracy 88 percent. Following are graphs of training and validation accuracy and training loss and validation los respectively.

Figure 13: Training and Validation loss

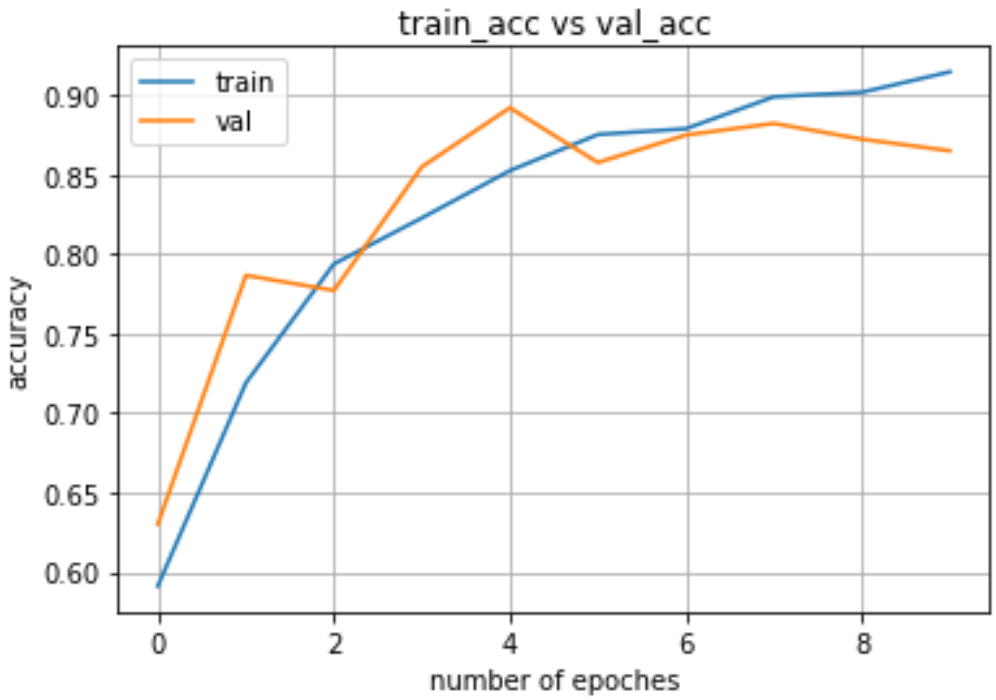


Figure 14: Training and Validation Accuracy

As mentioned above, we have a dataset of two different environments for product picked or unpicked classification, but only one environment dataset was used for training purpose while data of the other environment used for testing purpose on which the model has given very good results.

Figure 15: Results

Product Classification

Product Dataset Collection

We have collected dataset of four different products using mobile camera:

1. Bottle
2. Shampoo
3. Biscuits
4. Oil

First of all, we need to arrange these products in an enough quantity so we can shoot videos from different variations because more the data you best training you will get. So, had contacted to local store owner at Bun Hafiz Gee and requested him to give us these products so we can shot videos. Luckily the store owner was kind enough he gave us all the above-mentioned products and then get back and place these products in the shelves.

In the first iteration we have mounted mobile camera at left side of the shelf about an angle of 45° degree so that products can visible clearly while picking up from the shelve. We have shot 3 to 5 clips of videos each of length 2 minutes. In the first video clip only one person goes and picks up a product from the shelve, while in the second video more than one person used to to pick up product from the shelve.

Figure 16: Product Dataset

Similarly, in the second iteration we mounted the camera this time at the right side of the shelf and repeat the above-mentioned process of dataset collection.

Product Dataset Labeling

Dataset labeling or annotation is the way toward characterizing areas in a picture and giving them a literary mark for making ground truth. One of the numerous wonderful highlights of the people's perception is to make a rich portrayal of a scene inside a look at a picture. They can determine what are the items and their related traits in the picture. For instance, a picture can be characterized as "containing an individual, picking up bottle, oil, shampoo or biscuit from the shelf". Also, people can easily depict each article in the picture. During the previous not many decades, one of the major targets of computer vision research has been to reproduce this capacity, coming about in top to bottom investigations of computer vision issues remembering identification of items for a scene by portraying them utilizing their properties and making the ground truth.

OpenCV is the most widely recognized library for Computer Vision issues. We have extricated 6800 frames from the two recordings utilizing OpenCV. Subsequent to removing outlines from the recordings we marked these edges physically in bases of item picked is a bottle, oil, shampoo, or biscuit.

After extracting frames from videos using OpenCV we cropped product part from the whole frame, for this purpose we used Visual Object Tagging Tool (VOTT).

Then we divided our training and testing split 80% and 20%.

Figure 17: Cropped Product Dataset

Product Classification Using Keras

At this stage of our project we need to train a state-of-the-art CNN classifier using Keras that can classify the following store products:

1. Bottle
2. Shampoo
3. Biscuits
4. Oil

Our deep learning dataset consists of *6800 images* of the above-mentioned products like shampoo, bottles, oil, and biscuits. For this project, our goal is to train a convolutional neural network using Keras and deep learning that can classify the products with reasonable accuracy.

The products we will be classifying includes:

1. Bottle (1700 Images)
2. Shampoo (1700 Images)
3. Biscuits (1700 Images)
4. Oil (1700 Images)

We have diverse kind of dataset like we have collected it by mounting camera at different angels of store and at different light conditions so that, our model can generalize and classify these products. You will have obtained *82.9%* classification accuracy.

Products Model Architecture

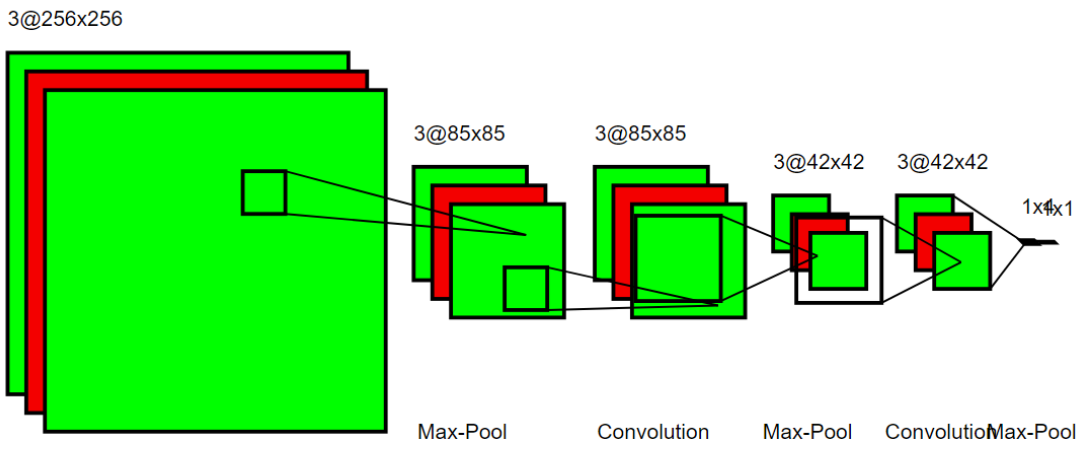


Figure 18: Model Architecture

Our CNN architecture is categorized by:

1. Using 3×3 convolutional layers stacked on top of each other in increasing depth (Feature Vectors) from 32 to 128.
2. Max polling that reduces the size of the feature vectors.
3. Dense Layers layers at the end of the network which uses SoftMax classifier because this is a multi-class problem.

Let's briefly discuss the model with code snippets.

```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
```

First of all, we import all the packages that are required for our model. These all imports are from keras.

Now, let's start discussing about the layers in our model:

```
# Layer one (input layer)
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape)) #
model.add(BatchNormalization(axis=chanDim))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
```

In the above code snippets, our first *conv => Relu => Pool* block.

This convolutional layer has *32* filters with a kernel of size 3×3. In We are using *Relu* activation function after batch normalization. Next for the purpose toreduce special dimensions we used pooling layer of size 3×3. This will reduce the dimensions from 256 × 256 to 32 × 32.

As should be obvious from the code square, we'll likewise be using dropout in our model architecture. Dropout works by haphazardly separating nodes from the current layer to the following layer. This procedure of irregular separates during training batches helps normally introduce redundancy into the model, not a single node in the layer is answerable for predicting a specific class, object, edge, or corner.

Now we will add 2 *Conv* and *Reul's* before Doing max pooling.

```
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(BatchNormalization(axis=chanDim))
model.add(Activation("relu"))
model.add(Conv2D(64, (3, 3),padding="same"))
model.add(BatchNormalization(axis=chanDim))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

This very general concept in deep learning that whenever we required richer feature set we always need to stack multiple *conv* and *Relu* layers together.

Now we can notice two things:

1. We have increased our filter size from 32 to 64. This is because as we move forward required smaller spatial dimension and more filters.
2. Secondly, we have decreased size of our max pooling layer from 3 × 3 to 2 × 2. we did this because want ensure that we do not need to reduce spatial dimensions too quickly.
3. Dropout is also performed at end of this code snippets.

Now we will again add 2 conv and Relu layers before pooling.

```
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(BatchNormalization(axis=chanDim))
model.add(Activation("relu"))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(BatchNormalization(axis=chanDim))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

At this stage you may have noticed that again we have increased filters size to 128. At last we have *Dense* layer with a *Relu* activation function as used above and *SoftMax* classifier is used because this multi-class problem.

```
# Dense layer
model.add(Dense(1024)) # 1024 OR 512
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
```

The completely associated layer is determined by Dense (1024) with a corrected direct unit enactment and bunch normalization.

Dropout is played out a last time — this time notice that we're dropping out half of the nodes during training. Ordinarily you'll utilize a dropout of 40-half in our completely associated layers and a dropout with a much lower rate, regularly 10-25% in past layers (if any dropout is applied whatsoever).

We balance the model with a SoftMax classifier that will restore the anticipated probabilities for each class name.

Model Accuracy and Loss

Looking at the graph we can see that our product classifier obtained:

1. *99.99%* product classification accuracy on training dataset.
2. *98.82%* product classification Validation accuracy on test dataset.

The training loss and accuracy graphs are given below:

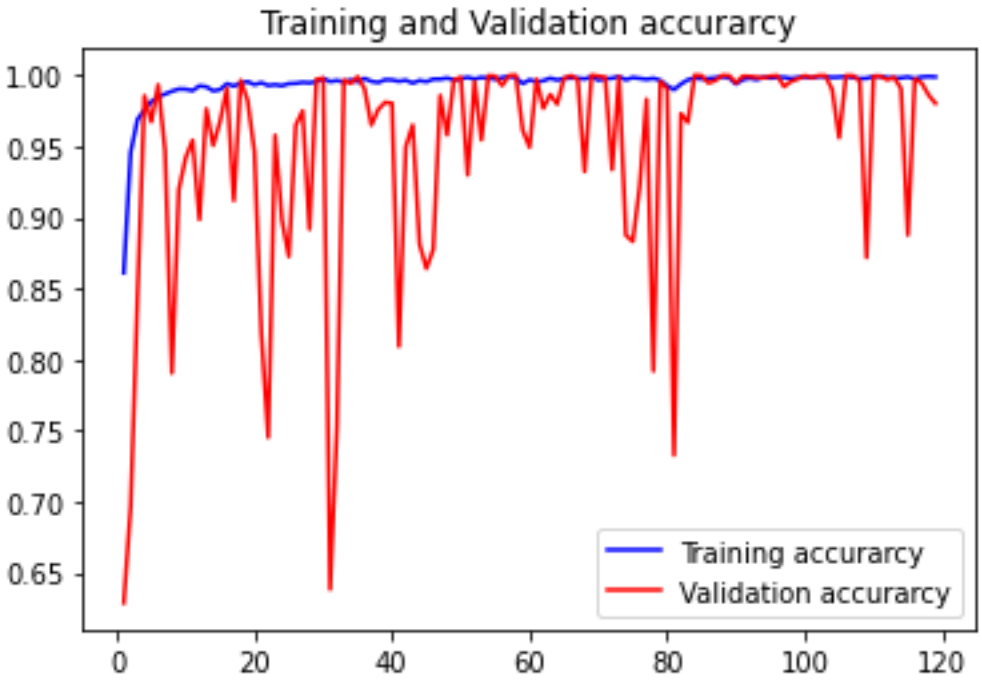


Figure 18: Model Accuracy

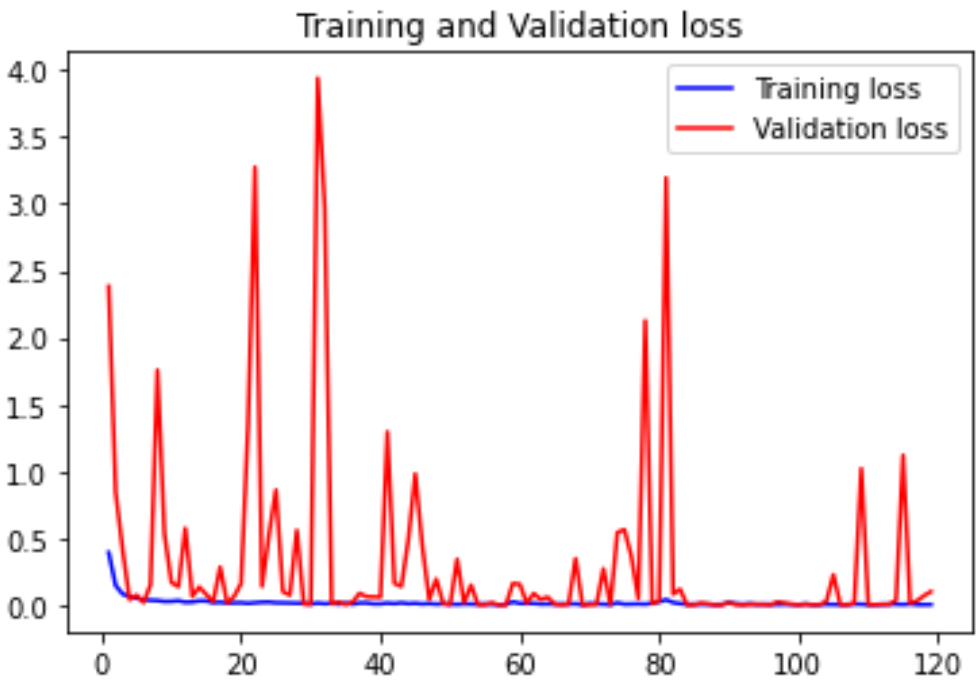


Figure 19: Model Loss

As mentioned in Keras API that if a training and validation accuracy fluctuates and does not increase in regular fashion this trend may cause overfitting of our model. On the other hand, if both training and validation accuracy increases in regular fashion this shows model is learning the data.

From the above given accuracy graph shows that training accuracy is increasing in at each epoch and also validation accuracy fluctuates somehow but in a long run it also increases which shows that our product classifier trained well.

Hand Localization

Hand Localization is still very challenging in the field of computer vision, several techniques has been proposed to localize hand in an image. But the most popular are Deep Learning and Pose Estimation.

There are several deep learning models are available on github for the purpose hand localization. Among all the models the best one is Tensor Flow Object Detection API which has the highest accuracy among all.

Hand Localization Using TensorFlow Object Detection API

Proposed When I have started work on hand localization for the very first time, I tried to use this API. I downloaded pertained weights of this API from the website and build it on my PC. This PAI works a little good when the hand is close to camera and clearly visible, but in other cases when the hand is far from the camera this API won't be able to localize the hand. So, as we have seen that in our dataset the individual picking up products from the shelves is very far from the camera and most of the time his hand is clearly visible So, that's why the result of this API on our dataset is not good. Below some results are attached for your reference.

Figure 20: TF Object Detection Api Results

The other popular approach of hand localization is posed estimation. By using the key points data returned by a pose estimation framework we could help to localize hand in an image. There are Three pose estimation frameworks are available on git, which are open source.

Pose Estimation

Human Pose estimation is an essential problem that has gained the attention of the Computer Vision community for the few decades. It is an important approach towards understanding people in images and videos. Human Pose Estimation is a process of the problem of localization of human joints (key points) in images or videos. It also leads as the search for a specific pose in space of all hinged poses.

Figure 21: Human Pose Estimation

Human Pose Estimation also has very cool applications. It is heavily used in Action recognition, Animation and Gaming. For instance, a very famous Deep Learning app Home-Court uses Pose Estimation to analyze Basketball player movements.

Classical approaches

The classical approach to hinged pose estimation is using the graphic structures framework. The basic idea here is to represent an object by a group of chunks arranged in a distortable layout. A chunk is an appearance sample which is matched in an image. Springs show the spatial connections between chunks. When chunks are specified by pixel location and orientation, the resulting structure can model conjugation which is very appropriate in pose estimation.

The above process comes with the restrictions of having a pose model not depending on image data. As a conclusion, research has focused on enhancing the descriptive power of the models.

Distortable part models: Yang and Ramanan use a mixture model of parts which indicates complex joint relationships. Distortable (Deformable) part models are a collection of samples arranged in a deformable arrangement and each model has global template and part templates. These samples are matched for in an image to recognize and detect an object. The Part based model

can model juncture well. This is however attained at the cost of limited expressiveness and does not take in global context into account.

Deep Learning based approaches

The classical channel has its restrictions and Pose estimation has been greatly reshaped by CNNs. With the start of Deep Pose by Toshev et al, research on human pose estimation began to move from classic approaches to Deep Learning. Most of the current pose estimation systems have worldwide embraced ConvNets as their main building block, largely replacement of hand-crafted features and graphical models. This approach has given in drastic advancements on standard benchmarks.

Alpha Pose Estimation

Alpha Pose Estimation is a famous top down approach of Pose Estimation. The top-down methods are based on the correctness of the person detector, as pose estimation is performed on the place where the person is pin pointed. Consequently, errors in localization and duplicate bounding box predictions can affect the pose extraction algorithm to perform insignificantly.

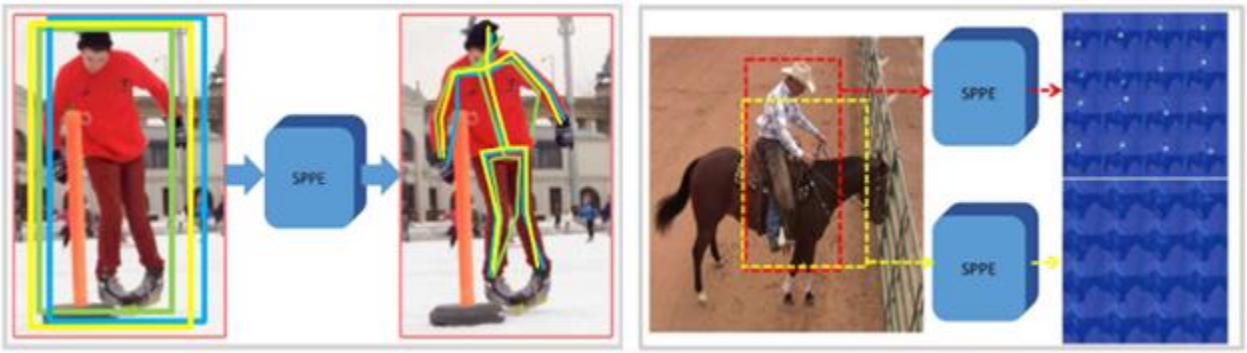


Figure 22: Alpha pose estimation

To settle this issue, the writer suggested the usage of Symmetric Spatial Transformer Network (SSTN) to draw out a high quality single person region from an imprecise bounding box. A Single Person Pose Estimator (SPPE) is pre-owned in this extracted region to evaluate the human pose skeleton for that person. A Spatial De-Transformer Network (SDTN) is used to rework the

estimated human pose back to the original image correlate system. In the end, a constant pose Non-Maximum Suppression (NMS) method is used to handle the issue of unnecessary pose deductions.

Moreover, the writers introduce a Pose Guided Proposals Generator to enlarge training templates that can better assist train the SPPE and SSTN networks. The important feature of RMPE is that this method can be expanded to any blend of a person detection algorithm and an SPPE.

Open Pose Estimation

Open Pose is one of the most famous bottom-up techniques for multi person human pose estimation and partially because of their well filed GitHub implementation.

As with many bottom-up approaches, Open Pose first notices parts (chunks) associated to every person in the image, heeding by allocating parts to distinct individuals. Following is the architecture of the OpenPose model.

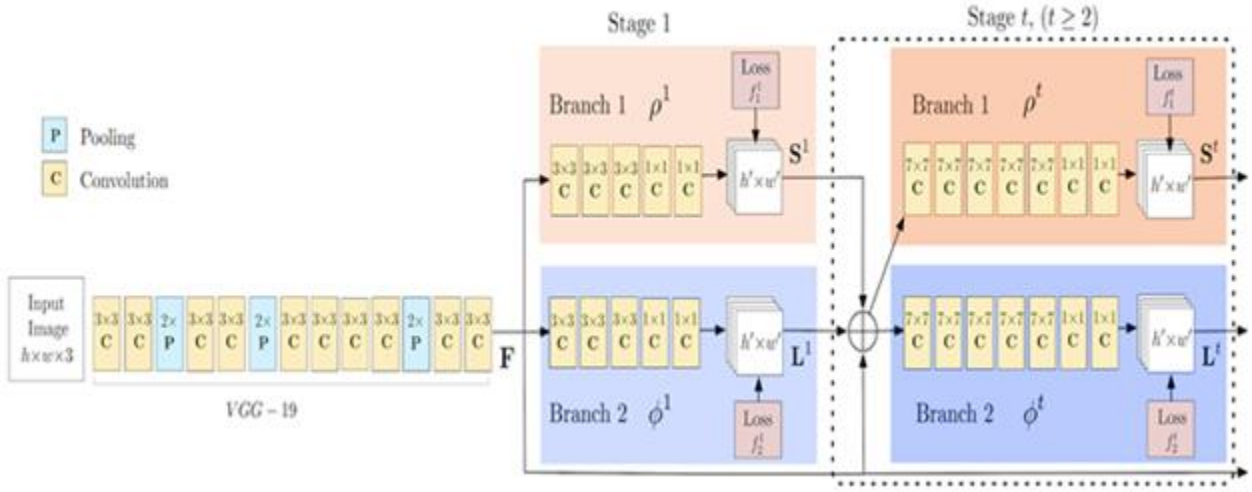


Figure 23: Open Pose Estimation

The Open Pose grid first draw out attributes from an image using the first few layers (VGG-19 in the above flowchart). The attributes are then sustained into two parallel branches of convolution layers. The first branch foresees a set of 18 confidence maps, with each map constitutes a particular part of the human pose skeleton. The second branch shows a set of 38 Part Affinity Fields (PAFs) which represents the degree of alliance between parts.

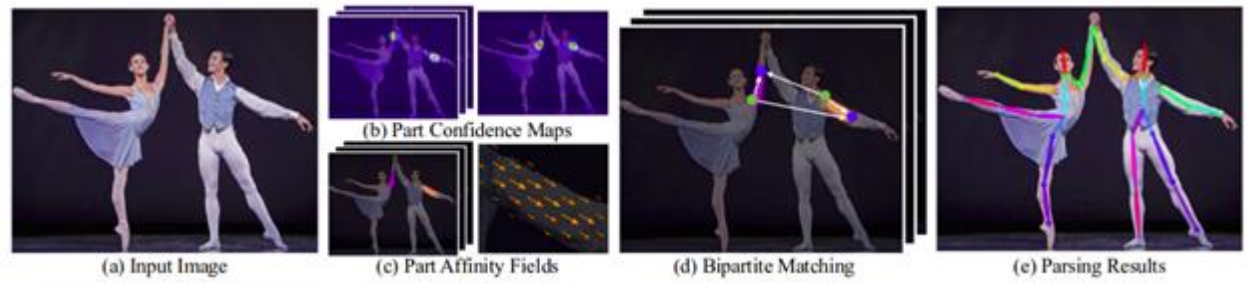


Figure 24: Open Pose process

Consecutive phases are used to refine the predictions made by each branch. Using the part confidence maps, multilateral graphs are set up between pairs of parts. Using the PAF values, weaker links in the multilateral graphs are cropped. Across the above steps, human pose skeletons can be estimated and allocated to every person in the image.

TF Pose Estimation

Open pose, human pose estimation algorithm has been executed using Tensor flow. It also furnished several forms that have some changes to the network structure for real time processing on the CPU or low power embedded devices.

So, above there is a briefe introduction about pose estimation frameworks. Now I will let you know I tried to use them for my project to localize hand.

Hand Localization Using TF Pose Estimation

TF pose estimation is light weight framework which can easily run on a cpu based machine. I had downloded it from github repository and downloaded all the prerequisites required to run this framework. Luckily I was succesful to run it on my cpu based machine. Then I tried to use my dataset to estmate pose of th person picking up products from the shelves. But the accuracy of this framework on my dataset is low. In 80% cases this frame won't even detect human in the frame, so how it could be able to estimate pose of a person. So this is the reason this I won't be able to use this framework.

Hand Localization Using Open Pose Estimation

This is one of the best pose estimation framework available on github repository. This gives you a very high accuracy in every condition no matter how bad your data is. First of all just for the purpose to test this frame work on my dataset I tried to build it on Colab run time environment, I was successful in doing this. This gives my very good results so next I need use hand key points to localize hand in an image which is the ultimate goal of using this framework.

Because I don't had GPU machine so I tried to use run time colab, But this I need to get the installation path of this framework on any machine, so that I can use that path with my python script which is able to get hand key points from this framework and draw bounding box around the hand.

But unluckily run time colab won't give me access to path were the Open Pose is installed that path was only be useful if you have credits on google cloud services. Next tried to get free credits by making a new account on google cloud which gave me 300hr free use of google cloud services. So, then I made a virtual Jupiter notebook with a support of GPU. But this won't create and returned a pop up message that you don't have GPU quota at any region. This my all struggle to use Open pose for the purpose of hand localization but won't be able to successful because I was not able to arrange a GPU based machine. So, therefore we left with the hand localization part of our project.

Technologies Used

TensorFlow is an open source software library for dataflow and differentiable programming covering a scope of works. It is a figurative math library and is used for machine learning applications such as neural networks. It is used for both exploration and manufacture at Google. TensorFlow was developed by the Google Brain team for intramural Google use.

Keras is an open source neural network library corresponding in Python. It is proficient in administrating on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano and PlaidML. Originated to enable fast research with deep neural networks, the center of attention is on being user friendly, modular, and extensible. It was developed as a module of the research effort of project ONEIROS.

Keras is a neural network library but TensorFlow is the open source library for a number of several tasks in machine learning. TensorFlow imparts both high level and low level APIs but Keras supports only high level APIs. Keras is built-in Python which moulds it way more user-friendly than TensorFlow.

OpenCV is a library of programming functions basically focused on real time computer vision. At the beginning it was developed by Intel, later it was supported by Willow Garage then Itseez. The library is cross platform and free for use below the open source BSD license.

Google Cloud Platform was propounded by Google, is a suite of cloud computing assistance that hastens on the identical framework that Google uses interiorly for its end user products, for example: Google Search, Gmail and YouTube.

Django is a Python based free and open source web infrastructure that accompanies the model template view prototype pattern. It is sustained by the Django Software Foundation, an American liberated organization settled as a 501 non-profit.

Django REST framework is a robust and pliable toolkit for building Web APIs. The Web explore-able API is an enormous serviceability win for your developers. Authentication policies include packages for OAuth1 and OAuth2. Serialization holds up both ORM and non-ORM data sources.

React is an open source JavaScript library for establishing user interfaces. It is prolonged by Facebook and a community of independent developers and companies. React can be used as a foundation in the development of single page or mobile applications.

Desktop Application

Functional Requirements

The functional requirements of Autonomous Checkout System are that it allowed the CCTV cameras to capture all movements in the form of images and videos so products and human's detection is possible to reduce the shoplifting of retailer. This system allows customer to buy product in the absence of shopkeeper. Autonomous Checkout System allows create an auto list of products which a customer has bought. This system also permits retailer to keep an eye on all procedure by viewing on online dashboard that what is happening inside all the business.

Non-Functional Requirements

Non-functional requirements of the project are below:

1. Performance

   How the automated checkout system responses to the request of customer determines the performance of the system. The system is efficient in resource utilization like fast response, exact visualization, storage etc.

2. Scalability

   The system provides different features for both admin and customer. The system is able to serve large number of customers as per demand.

3. Reliability

   The system provides a lot of variety of products in one place for the consumer. Products and prices are made affordable for middle class and upper class in society. The

system provides a well-designed interface that is comprehensible and controllable for admin, helping admin to complete his work successfully and efficiently, and to feel competent and satisfied. Effective interfaces are designed that is reliable to use. The system is highly useable for admin and customer.

4. Interoperability

The system is built by integrating python using Django Framework, python, html, css, bootstrap, JavaScript. So, they operate together.

Feasibility Analysis

1. Operational feasibility

The system is highly user friendly and it is much easier to interact with the customer. Customer does not need special training to operate the system. However, the system will provide maximum ease.

2. Technical feasibility

This is concerned with the specifying equipment and software that successfully satisfy the requirements. The system is technically feasible as it can be developed easily with the help of available technology. The system requires HTML, CSS, Bootstrap, JavaScript which is used as front-end, Django and Django Rest framework as back-end and python as both.

3. Economic feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of the system. The tangible benefits proposed that the manual work and burden is reduced maximum as possible, resulting the reduction in manpower requirement and cost incurred on manpower as well. The system provides many benefits that can't be measured in terms of money for e.g. user friendliness, more efficient user response, maintenance of database, etc.

Use Case Diagram

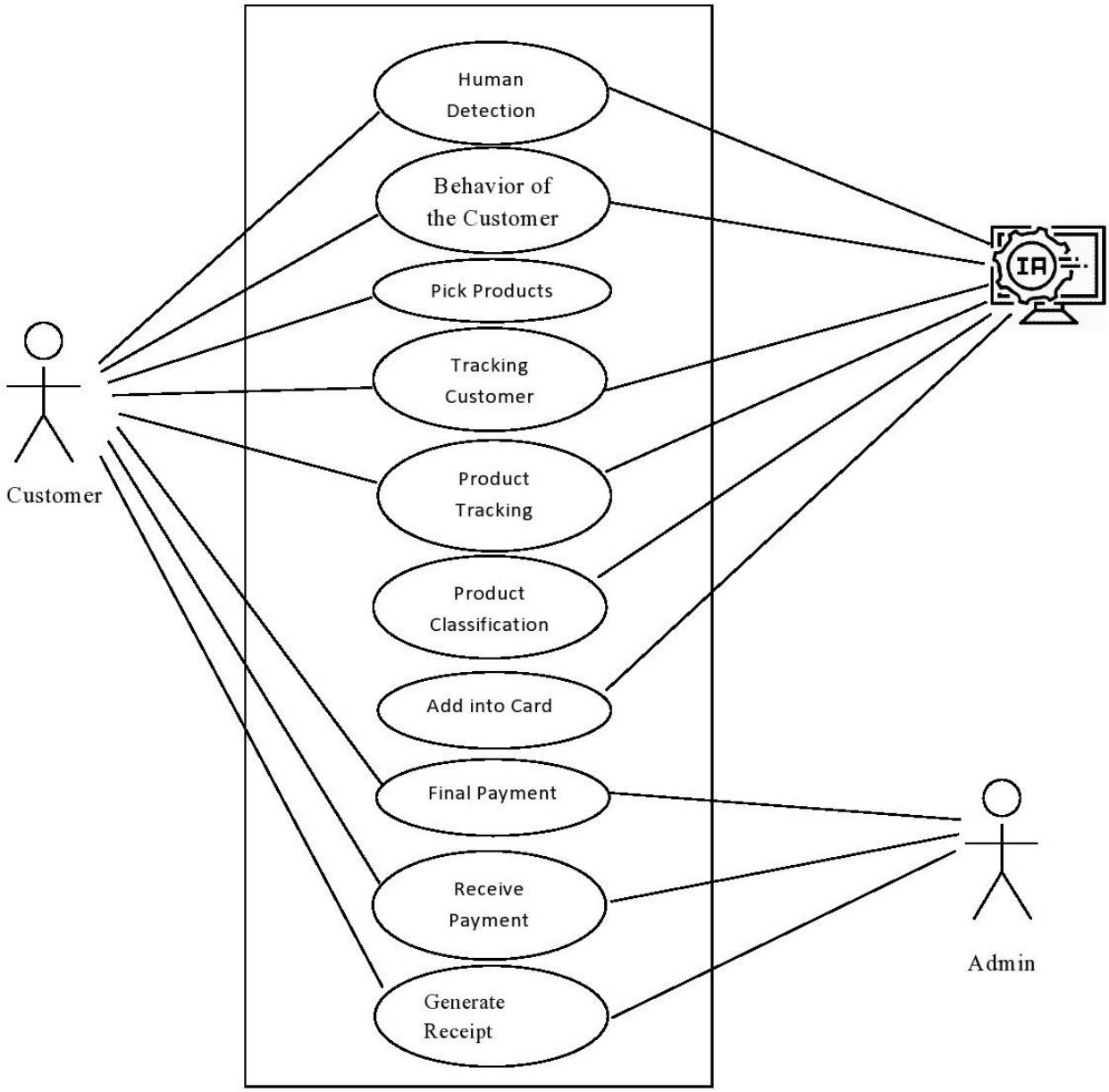


Figure 25: Use case diagram for Autonomous Checkout System

Figure No 0.0 shows the use case diagram for this autonomous checkout system. User, Admin and AI based system are the three main part that are include in Autonomous Checkout System where Admin is known as shop owner or super user. AI system that is handling human detection, behavior of the customer, product and customer tracking and finally classification of the

product and add into the customer cart. User can pick the one or multiple products and pay payment which products he/she picked and receive a receipt. Admin can also act as user so he/she can do the tasks of users and also receive the payment and generate a receipt for the customer.

Use Cases:

- Customer visits shopping center
- He may buy item or just visits
- Detection of customer through Mobile Net model
- Checking the behavior of the customer
- Picking the product for purchasing
- Tracking the customer
- Tracking the Products
- Classify the products
- Add into the customer cart

  The process can be repeated for more items
- Final payments
- Final cart view on admin dashboard
- Admin Receive the payment
- Generate a final receipt

E-R diagram

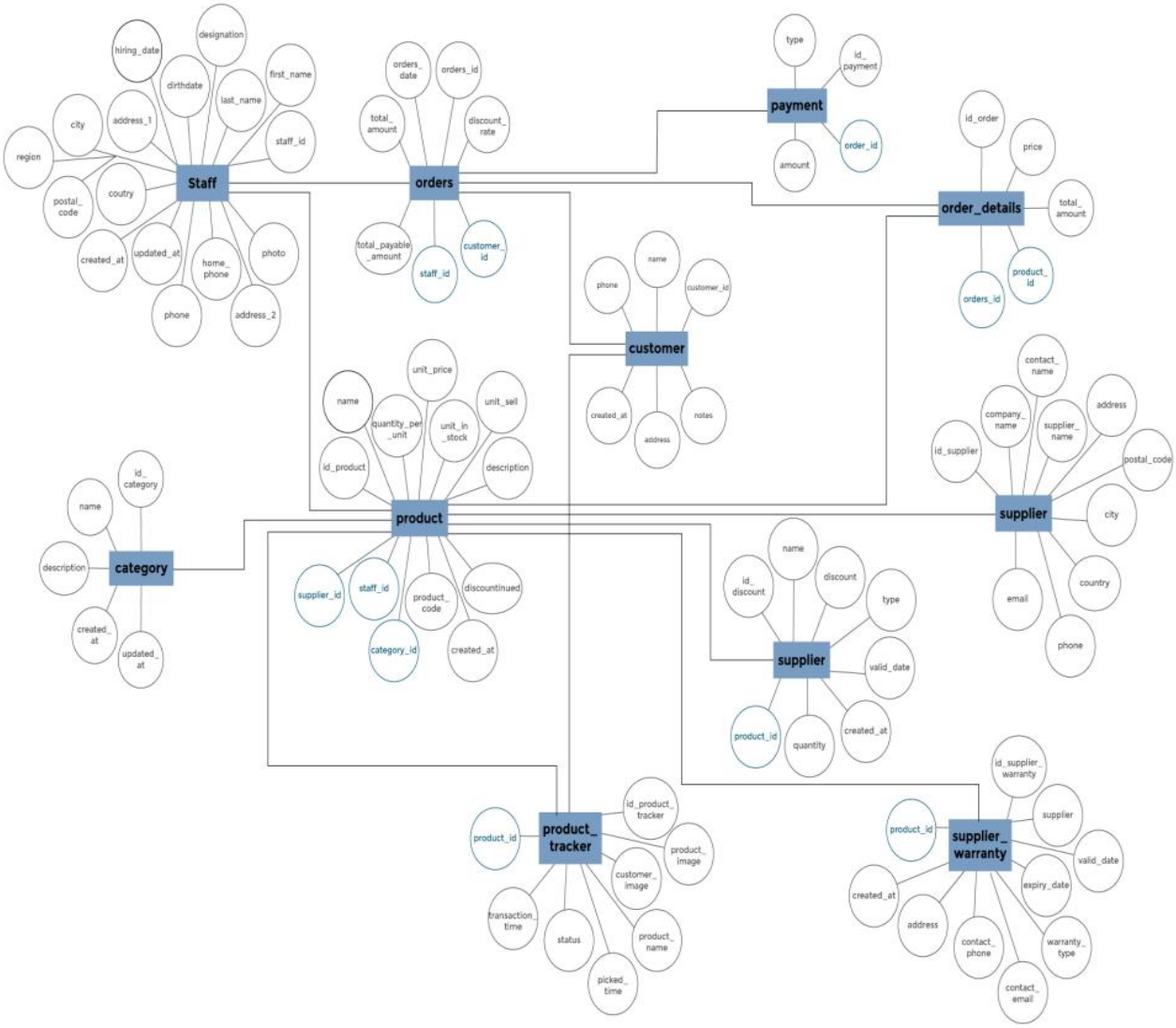


Figure 26: E-R Diagram for Autonomous Checkout System

Figure No 0.0 shows the logical relationship between all the entities that are including in Autonomous checkout system. Staff, Products, Suppliers, Orders, Products Tracker, and Customer are the entities and each entity has their respective attributes.

E-E-R- Diagram/Schema Diagram

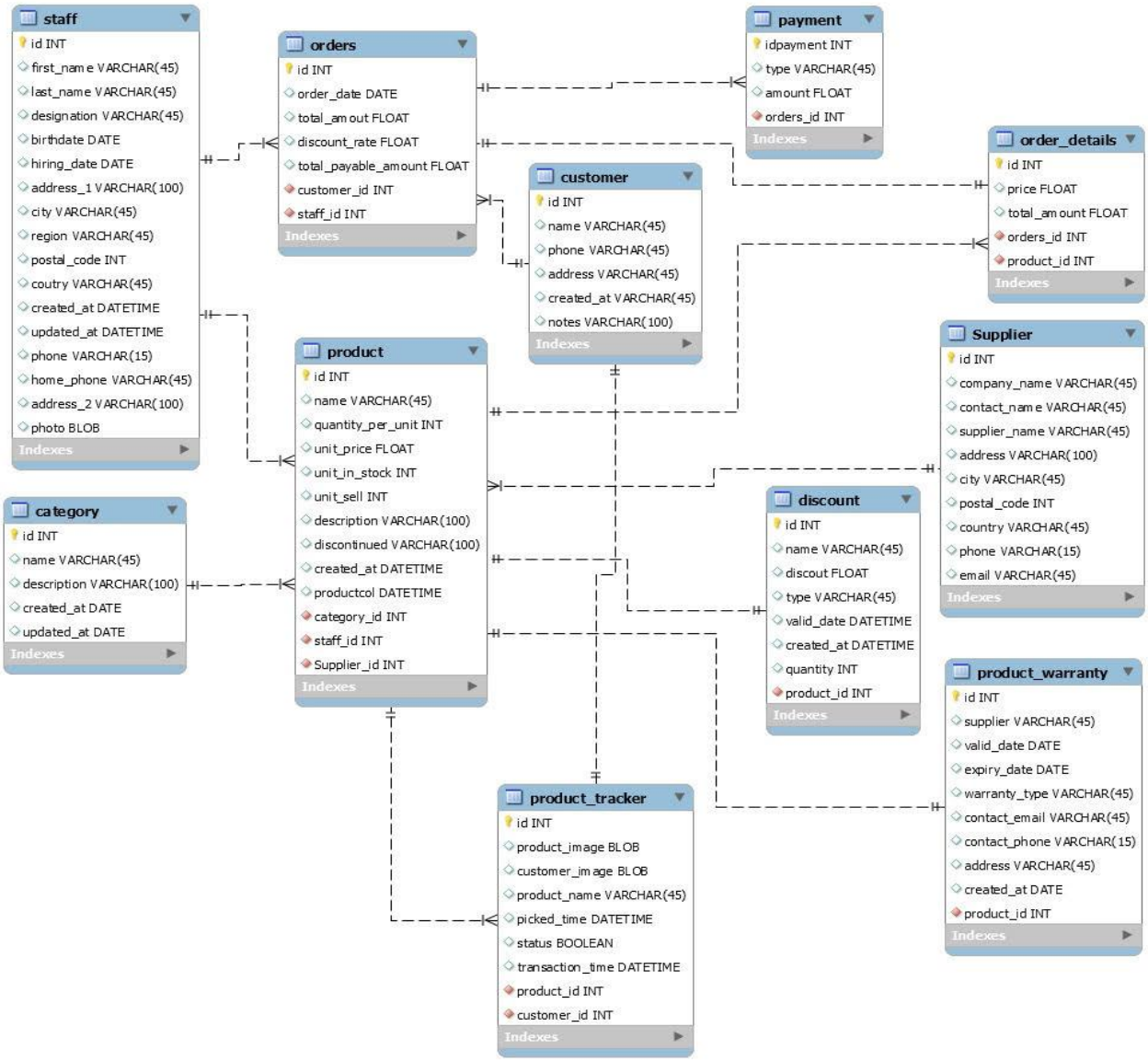


Figure 27: EE-R Diagram for Autonomous Checkout System

The Figure No 0.0 is the complete visual representation of organization and structure of Autonomous Checkout System database. It contains schema objects like tables, columns, data types, relationships, primary key, foreign keys, etc. It shows eleven tables, along with their data types, relationships between the tables, as well their primary keys and foreign keys.

Class Diagram

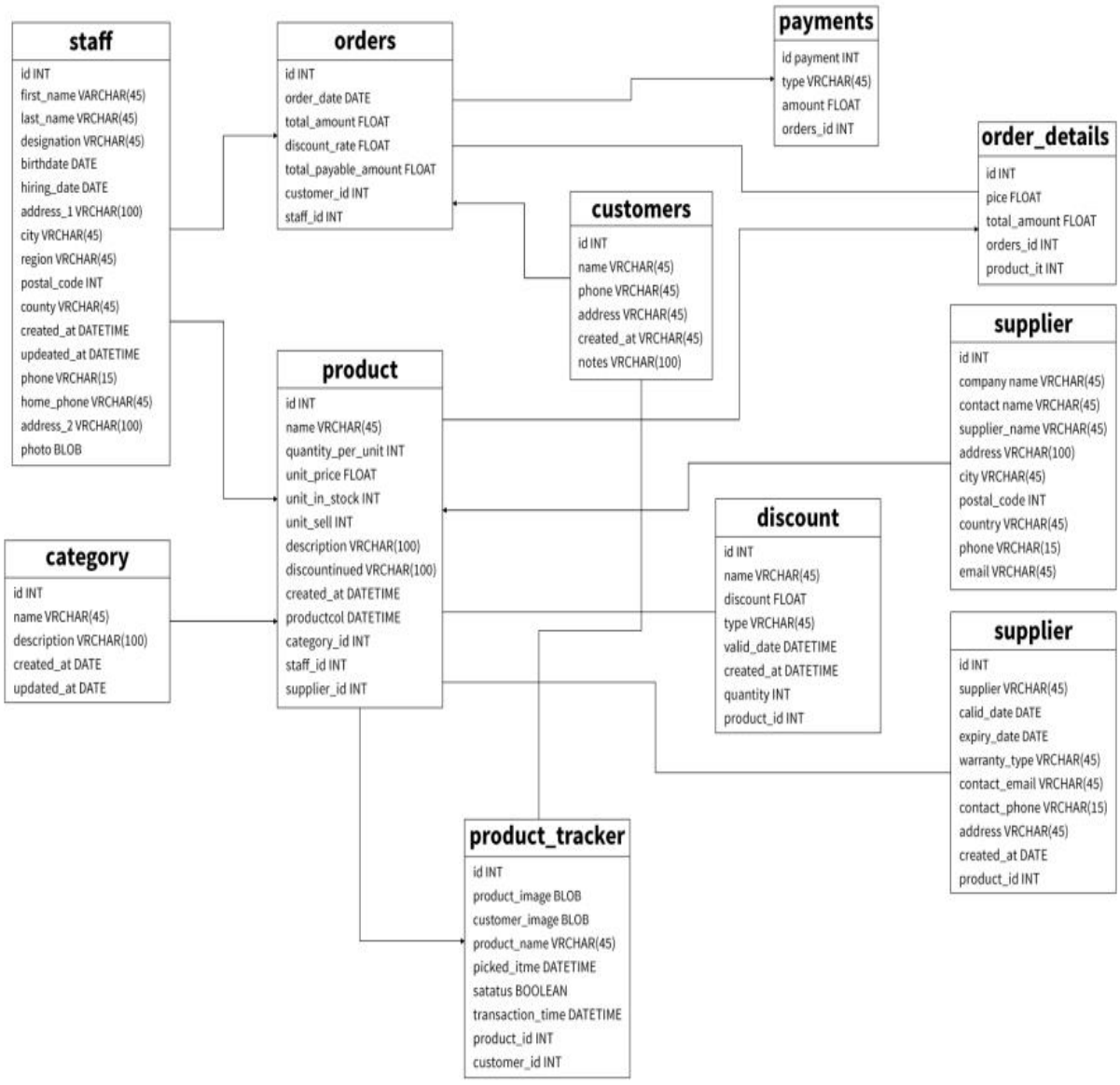


Figure 28: Class Diagram for Autonomous Checkout System

The Figure 0.0 is a complete structure diagram that describe the structure of Autonomous Checkout system by showing the system's classes, their attributes, Methods and the relationship between the all classes objects. In classes diagram are represented with boxes that contains two

divisions. The top box contains the name of the class. The second box contains the attributes or characteristics of the classes.

METHODOLOGY

Tools Used

1. Front-end Tools
    a. HTML5 with Jinja Templates
    b. CSS4
    c. Java Script
    d. jQuery
2. Back-end Tools
    a. Python with Django Framework
    b. Django Rest Framework for API's
    c. PostgreSQL
    d. PGAdmin 4 backend server for Database

Admin and Customer

The system has been implemented using the Python Programming Language with Django Framework. The system deals with the detection of customer. The system checks the behavior of customer. Customer picks the product of his choice. The system tracks the product and the customer. Products are classified. Customer adds his selected product in cart. He pays final payment. Final payment is received by admin and generated a receipt to customer.

In the Admin and Customer side, for each feature, there are classes, models, functions or Python codes in web pages involving the requests from users or display the result on the desktop application.

Admin Side

- Manage Product Category (Create, Update, Delete)
- Category model is created using "class Category (models.Model)".
- Attributes fors categories are defined under the Category class.
- Using make migration module the class is migrated to database
- From data base a new category can be added, existing category can be updated or deleted.
- Manage Product (Create, Update, Delete)
- Model for product is created using "class Product (models.Model)"
- Attributes of product are defined under the Product class.s
- Using make migration module the class is migrated to database
- From data base a new product can be added, existing product can be updated or deleted.

- Manage Customer
- The new customers and their status are viewed by using Django User model.
- New customers are saved in database through registration process using the function def register_user (request):
- Django User model facilitates control users.
- Manage Order (Orders are stored and can be viewed from Orders Model)
- Class Order (models.Model) is used to build Orders Model along with defining the attributes of the class.
- Manage Tracking
- Admin can manage the tracking of product and customer.
- Manage Final payment
- Admin will receive final payment.
- Generate receipt
- Admin will generate receipt after final payment.

Customer Side

- Pick product
- Any customer can pick the product from the rack from the front of the camera in utility store.
- Customer can easily buy any product from the rack.
- Pays final payment
- Customer pays final payment after picking products from racks.
- After paying final payment, customer gets receipt of his purchasing.

IMPLEMENTATION AND TESTING

Database

A database is an arranged collection of data, usually stored and accessed electronically from a computer system. Databases are more complex, they are generally developed using formal design and modeling techniques. PostgreSQL is a free and open-source relational database management system emphasizing extensibility and SQL compliance. PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments and help to manage your data no matter how big or small the dataset.

Only PostgreSQL supplies enterprise-class performance and functions amidst current Open Source DBMS with no end of development possibilities. One of the attrubutes of PostgreSQL is that there is a wide diversity of communities. In respect of PostgreSQL as Open Source DBMS, users themselves can develop modules and propose the module to the group.

PGAdmin is a administrative tool for PostgreSQL and derivative relational databases. It can be run either as a web or desktop application. The pgAdmin 4 client attributes a highly customizable display that features drag and drop dashboard that you can arrange to make the best use of your desktop environment. In a Server Deployment, the pgAdmin application is deployed in back of a webserver or with the WSGI interface. In a Desktop Deployment, the pgAdmin application is configured to use the desktop runtime environment to host the program on a platform.

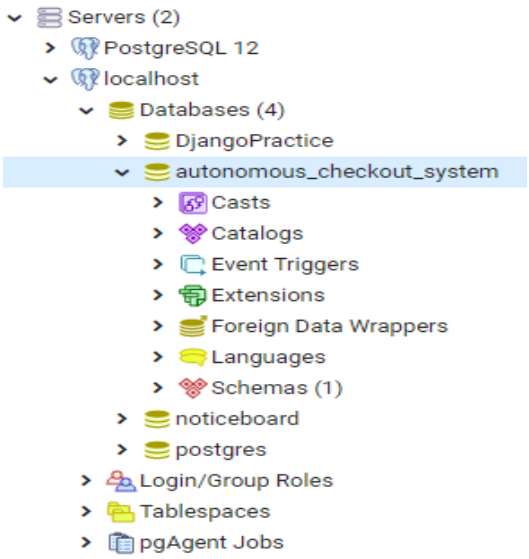


Figure 29: Database in PostgreSQL

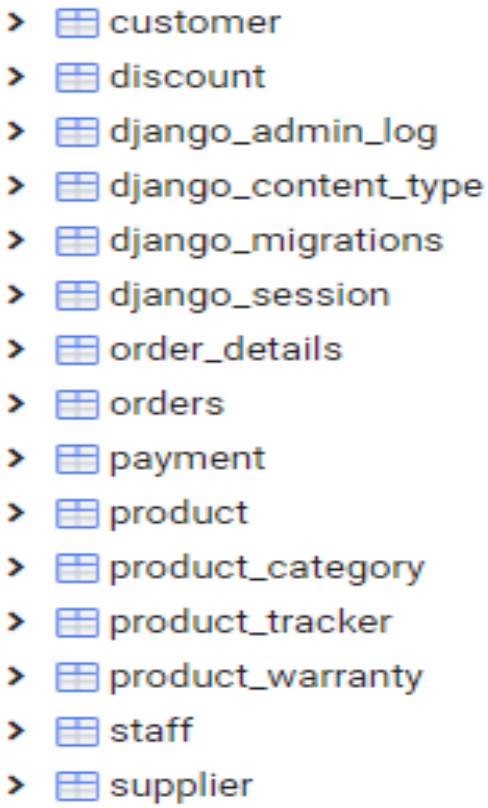


Figure 30: Tables in Database (Autonomous Checkout System)

Front-end design

Frontend design deals with creating the HTML, CSS, and presentational JavaScript code that makes up a user interface.

It defines all the options that are available to admin to manage all the activities including staff management, customer management, product management, payment management, supplier management, product tracking, product warranty etc.

Figure 31: Menu Bar for Admin Dashboard

Live video demonstrates the live visualization of purchasing system.

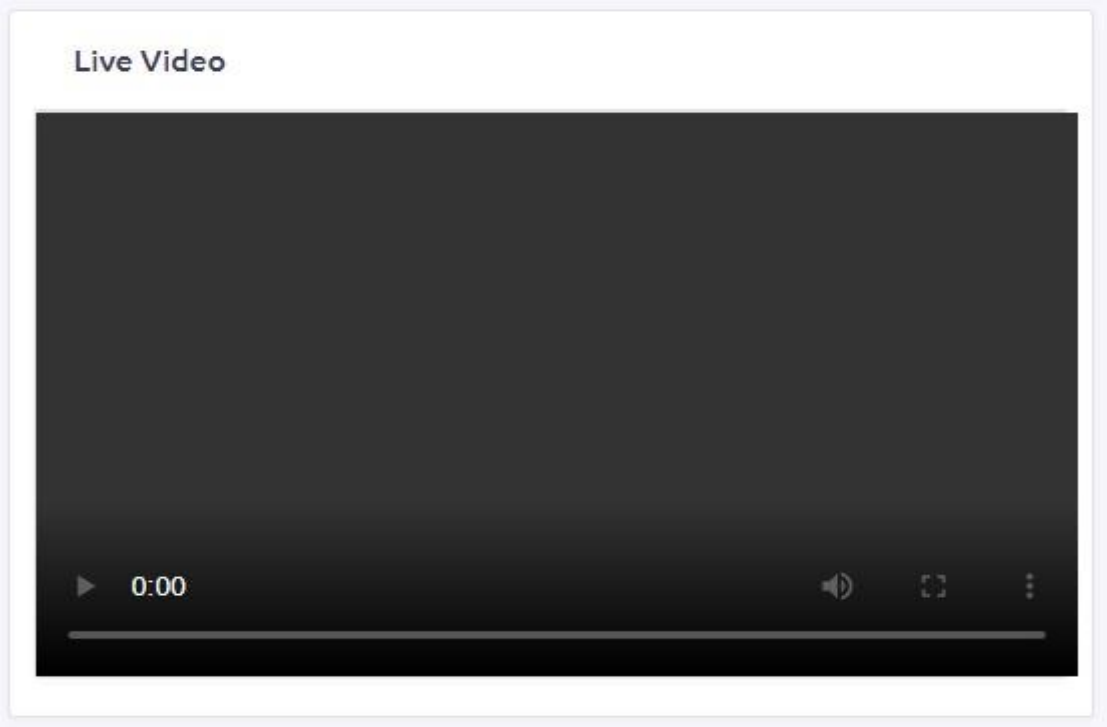


Figure 32: live visualization screen

It shows all the active customers who are purchasing products.

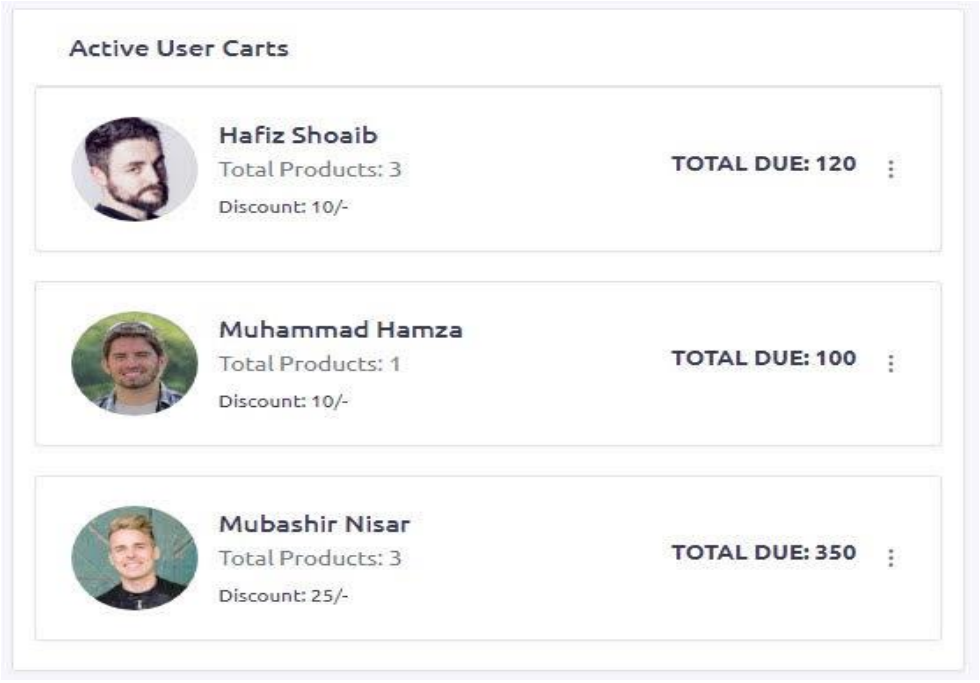


Figure 33: Active Customers

This part of dashboard consists of four sub parts.

- Sales:
  It depicts all the sales history.
- Customers:
  It represented number of customers in form of graph.
- Active Customers:
  This part presents the active customers who are currently engaged in shopping.
- Revenue:
  It shows total that has been earned.

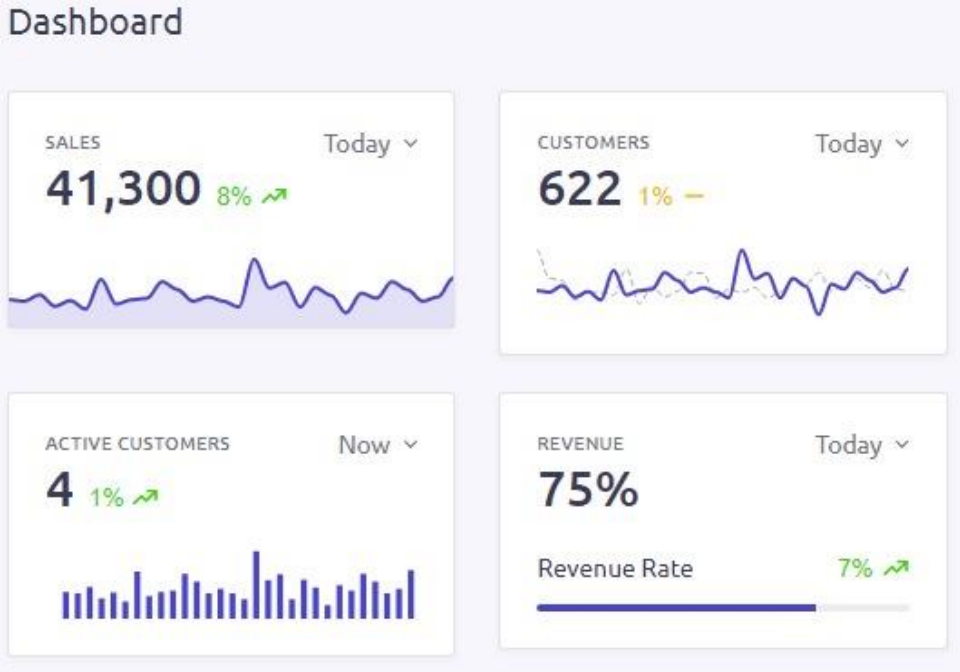


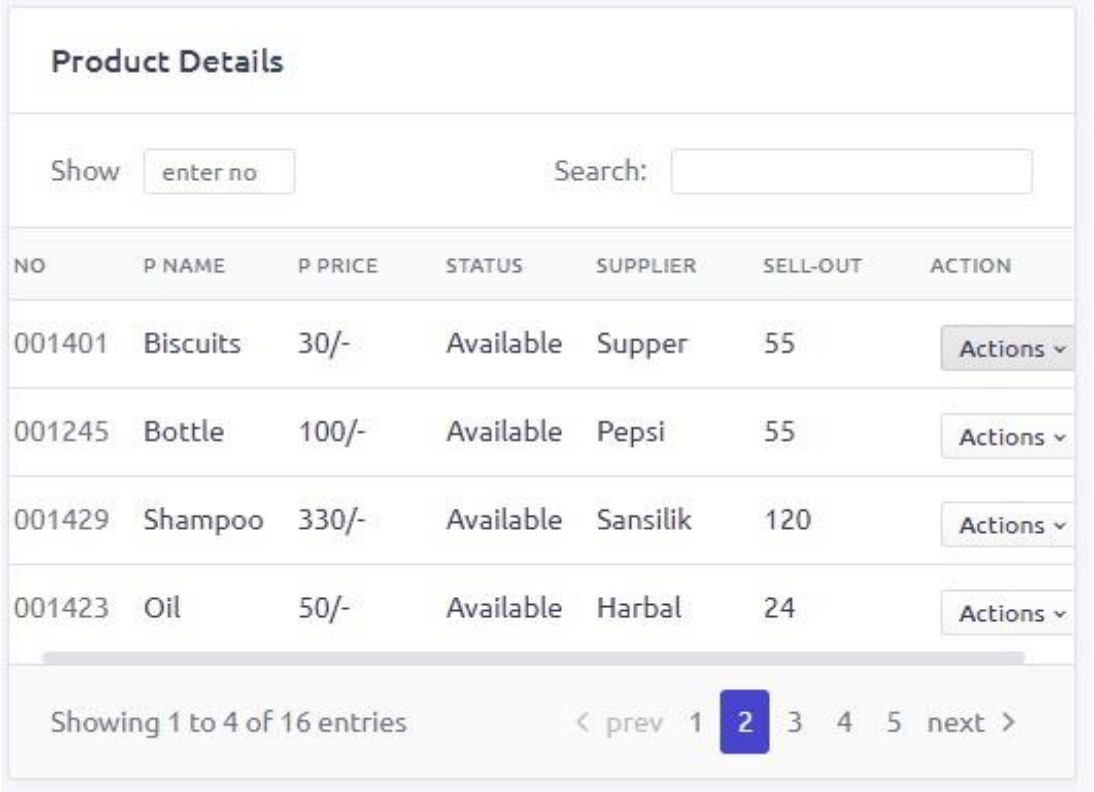
Figure 34: Admin Dashboard

Profit depicts the excess of revenue over cost.

It is of admin's benefit in business.

Suppliers box shows the number of suppliers to business and their details that which company to they belong, how many products they supply and either they are in action or not.

Product Details give details about the products names, products prices, status, supplier, number of sell out products and actions that are performed on products.



Back-End Design

These are Django Rest API's for desktop application. These are connected with PostgreSQL database server. Django REST framework is a robust and pliable toolkit for building Web APIs. The Web explore-able API is enormous serviceability wins for your developers. Authentication policies including are packages for OAuth1 and OAuth2. Serialization is that holds up both ORM and non-ORM data sources.

The back-end is basically how the site works, updates, and changes. This refers to everything the user can't see in the browser, like databases and servers.

This back-end design of API root defines the product tracking.

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "producttracking": "http://127.0.0.1:8000/api/v1/product/product/"
}
```

It demonstrates the order of product, details of order and final payment.

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "order": "http://127.0.0.1:8000/api/v1/order/order/",
    "orderdetail": "http://127.0.0.1:8000/api/v1/order/orderdetail/",
    "payment": "http://127.0.0.1:8000/api/v1/order/payment/"
}
```

It shows the category of product, the exact product, warranty details of product and discount over product.

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "productcategory": "http://127.0.0.1:8000/api/v1/product/productcategory/",
    "product": "http://127.0.0.1:8000/api/v1/product/product/",
    "warranty": "http://127.0.0.1:8000/api/v1/product/warranty/",
    "discount": "http://127.0.0.1:8000/api/v1/product/discount/"
}
```

It represents details of staff, customers and suppliers. It means that all the subjective details are given below:

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "staff": "http://127.0.0.1:8000/api/v1/user/staff/",
    "customer": "http://127.0.0.1:8000/api/v1/user/customer/",
    "supplier": "http://127.0.0.1:8000/api/v1/user/supplier/"
}
```

References

[1] Khan, M., Awais, M., Shamail, S. and Awan, I. (2011). An empirical study of modeling self-management capabilities in autonomic systems using case-based reasoning. Simulation Modelling Practice and Theory, 19(10), pp.2256-2275.

[2] Lecun, Y, Bottou, L, Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[3] LV, J. (2019). Visual Attentional Network and Learning Method for Object Search and Recognition. Journal of Mechanical Engineering, 55(11), p.123.

[4] Marino, D., Ireifej, S. and Loughin, C. (2011). Micro Total Hip Replacement in Dogs and Cats. Veterinary Surgery, 41(1), pp.121-129.

[5] Radovic, M., Adarkwa, O. and Wang, Q. (2017). Object Recognition in Aerial Images Using Convolutional Neural Networks. Journal of Imaging, 3(2), p.21.

[6] Ren, S., He, K., Girshick, R. and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), pp.1137-1149.

[7] Datasciencemasters.org. (2020). The Open Source Data Science Masters by datasciencemasters. [online] Available at: http://datasciencemasters.org/ [Accessed 25 Feb. 2020].

[8] Thika, R. and lakshmi, S. (2017). Object Detection and Semantic Segmentation using Neural Networks. International Journal of Computer Trends and Technology, 47(2), pp.95-100.

[9] Triana, E. and Pasnak, R. (1981). Object permanence in cats and dogs. Animal Learning Behavior, 9(1), pp.135-139.

[10] Youtube.com. (2020). YouTube. [online] Available at: https://www.youtube.com/ [Accessed 25 Feb. 2020].

[11] Agarwal A, Triggs B (2006) Recovering 3d human pose from monocular images. IEEE Trans Pattern Anal Mach Intell 28(1):44–58

[12] Eichner M, Ferrari V (2010) We are family: joint pose estimation of multiple persons. In: European conference on computer vision (ECCV), Heraklion

[13] Django Framework: https://docs.djangoproject.com/en/3.0/

[14] Django Rest Framework: https://www.django-rest-framework.org/

[15] ReactJS, JavaScript Library: https://reactjs.org/docs/getting-started.html