SMART VENDING MACHINE

By

Devashish Krishna (RA2011031010062)

Riya Singh (RA2011031010044)

Aditi Mishra(RA2011031010041)

Under the guidance of

Dr. V.R Balasaraswathi

In partial fulfilment for the Course

of

18CSC305J – ARTIFICIAL INTELLIGENCE

in Computer Science Engineering Specialization in IT

FACULTY OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Katankulathur, Chengalpatu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)BONAFIDE

CERTIFICATE

Certified that this mini project report "**Smart Vending Machine**" is the bonafide work of **Devashish Krishna (RA2011031010062), Riya singh (RA2011031010044) , Aditi Mishna (RA2011031010041)** who carried out the project work under my supervision.

SIGNATURE
Dr. V.R Balasaraswathi (Professor NWC) SRM

Institute of Science and Technology

TABLE OF CONTENTS

Chapter 1. Abstract

A vending machine is an automated machine that provides items such as snacks, beverages, cigarettes, and lottery tickets to consumers after cash, a credit card, or other forms of payment are inserted into the machine or otherwise made. Vending Machines with digital payments, apps for ordering, and cloud-connected systems for real-time monitoring are called Smart vending machines. They deliver products without the need for human intervention.

This paper proposes a smart vending machine system combined with deep learning and machine learning technologies. The proposed system is combined with temperature and camera sensor to obtain consumer without individual information and upload this information to cloud server. The system uses face recognition with deep learning to obtain the gender information. It uses the k nearest neighbors (KNN) machine learning method to group based on temperature, time, price and gender information. The proposed system relies on grouping information to dynamically adjust price in real time.

The model of the vending machine successfully detects the coin and dispenses liquid output. It is also proved to be customer friendly. The system developed in this research work can be used as a base for future development .Vending machine operators provide and service automated machines that sell merchandise, primarily snack foods and soft drinks, but also cigarettes, newspapers and other goods. This industry does not include revenue from soft drink producers that operate their own vending machines.

Chapter 2. Introduction

Smart vending machines are the next evolution of retail because they allow businesses to reach customers in new locations and in new ways. These machines work around the clock. They also serve customers with ease and efficiency by only displaying what selections are in stock. Businesses no longer need to invest in expensive packaging.

To implement Smart vending Machine various object detection libraries and frameworks are available, and JavaScript provides a convenient platform for building web-based applications. This JavaScript code provides a framework for building a Smart vending machine web application that uses a device's camera to capture video streams

The code provides a foundation for implementing various product detection applications, including surveillance systems, traffic monitoring, and wildlife tracking. Additionally, the code is user-friendly, customizable, and provides real-time alerts and notifications for identified objects, making it suitable fora wide range of applications.

In summary, A vending machine is a system that dispenses products stored inside it in exchange for coins or tokens. A smart vending machine is an automated machine which dispenses products and engages consumers with video, audio, fragrance, touchscreen controls, cashless payment, and gesture-based communication.

Problem Statement : -

The problem statement that can be derived from this code is how to improve the Smart vending Machine algorithm to achieve beter accuracy and performance. This can involve exploring different types of hand detection , fine-tuning the existing model, optimizing the code for faster execution, and implementing additional features such as object tracking or classification. The goal is to create an efficient and reliable vending system that can be used in various applications such as security, robotics, and autonomous driving.

Objective : -

- Identify objects accurately in an image or video.
- Achieve high accuracy with low false positive and false negative rates.
- Enable automation and enhance safety in various applications, such as self-driving cars andsurveillance systems.
- Improve disease detection and diagnosis in medical imaging.

Count objects, track their movements, and predict their future positions for beter decision-making.

- Detect and identify anomalies.

Chapter 3. LITERATURE SURVEY

Prof. Preetilatha proposed microcontroller based vending machine. It support cashless payment as the input by the scanning of RFID card and dispenses produces like A4 sheets, pencil, pen, eraser, etc. Thus it can be supportive to trade stationary items automatic. Prof. Kamal Nathan proposed microcontroller based automatic paper vending machine.[7] It is proposed to convey the paper to the published by utilizing the sensors and dependent on the Mechatronics standards. It acknowledges coins as an input and dispenses sheets as a yield.

The software used is “embedded”.
Hence it very well may be useful for school and school. Dr. Aman Qureshi proposed FPGA based vending machine which supports four products and two coins. It accepts coins as input in any sequence and dispenses product when required amount is deposited and returns the change if entered amount is greater than the prize of product. It also supports cancel feature through which a user can withdraw the request any time and enter money will be returned back. The algorithm is implemented in Verilog HDL and design is implemented on Xilinx Sparten-3 XC3S400FPGA.

This machine accepts money as an input to despense the products and returns back the money without dispensing the product to the customer if the product is out of date. Therefore it can be advantageous to confirm the good quality of the product together with amount and cost. Dr. Preeti Sharma proposed a Reverse Vending Machine based on FPGA and in this paper the author describes FPGA based vending machine. Publishers B.V.N.R.Siva Kumar, Kurisetti.Siddhartha Roy,C Bapaiah Naidu, Addanki John, use of IR sensor for ship dispenses or not dispenses confirmation.

If the ship is detected by the sensor, it sends a signal to the Arduino nano for opening of the bridge and if the ship is not detected by the sensor, it sends the signal to the Arduino nano for closing the bridge.[8] Publishers Sooraj, Bony Mons, Dr.Jisna Kuruvilla in IOT (internet of things) Based Vending Machine with Cashless Payment Conducted So Far: Cashless Payment System in Vending Machine using IOT. A vending machine is an automated machine that provides items such as snacks to employees; a card is inserted into the machine. This is cashless payment concept.[9]

Existing Methods

ResNet

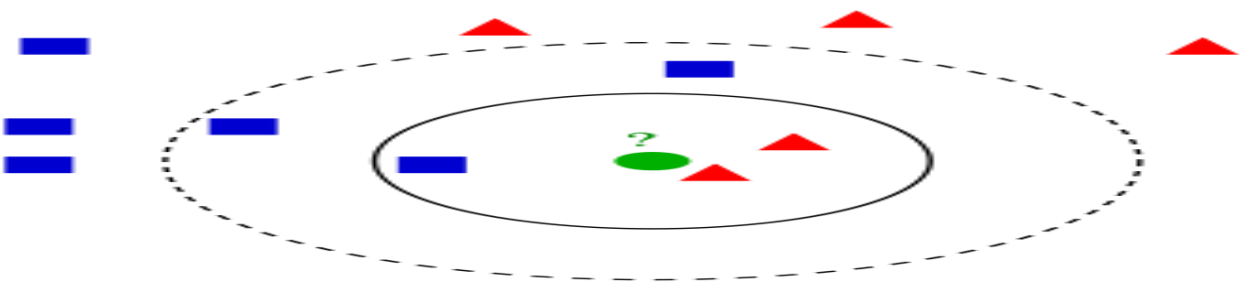
To train the network model in a more effective manner, we herein adopt the same strategy as thatused for DSSD (the performance of the residual network is better than that of the VGG network). The goal is to improve accuracy. However, the first implemented for the modification was the replacement of the VGG network which is used in the original SSD with ResNet. We will also add a series of convolution feature layers at the end of the underlying network. These feature layers will gradually be reduced in size that allowed prediction of the detection results on multiple scales. When the input size is given as 300 and 320, although the ResNet–101 layer is deeper than the VGG–16 layer, it is experimentally known that it replaces the SSD's underlying convolution network with a residual network, and it does not improve its accuracy but rather decreases it.

R-KNN

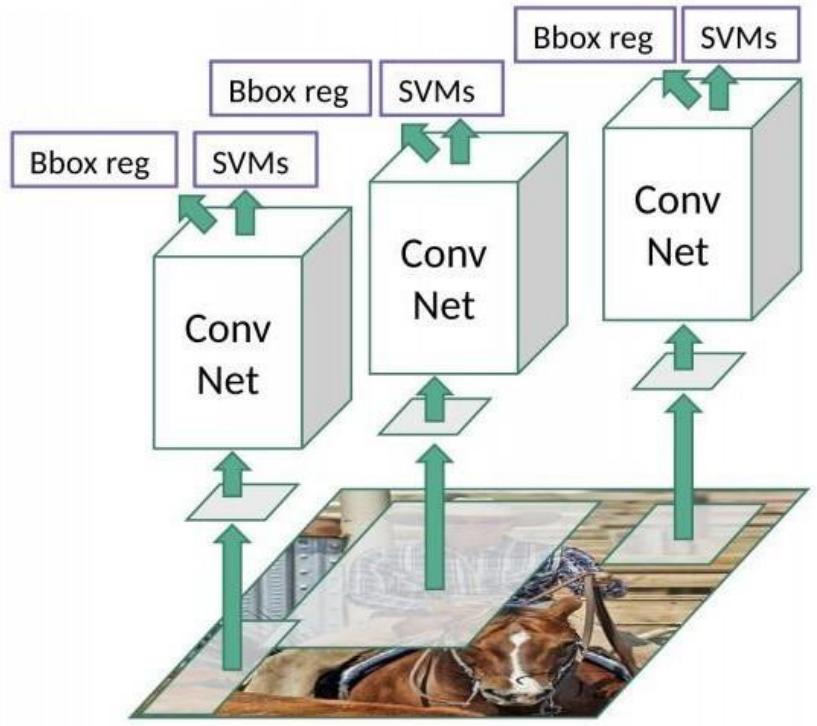
K-Nearest Neighbors Algorithm. The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Selective Search:

1. Generate the initial sub-segmentation, we generate many candidate regions
2. Use the greedy algorithm to recursively combine similar regions into larger ones
3. Use generated regions to produce the final candidate region proposals



These 2000 candidate regions which are proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The KNN plays a role of feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM for the classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values for increasing the precision of the bounding box. For example, given the region proposal, the algorithm might have predicted the presence of a person but the face of that person within that region proposal could have been cut in half. Therefore, the offset values which is given help in adjusting the bounding box of the region proposal.



Problems with R-KNN

K-Nearest Neighbor or K-NN is a Supervised Non-linear classification algorithm. K-NN is a Non-parametric algorithm i.e it doesn't make any assumption about underlying data or its distribution.

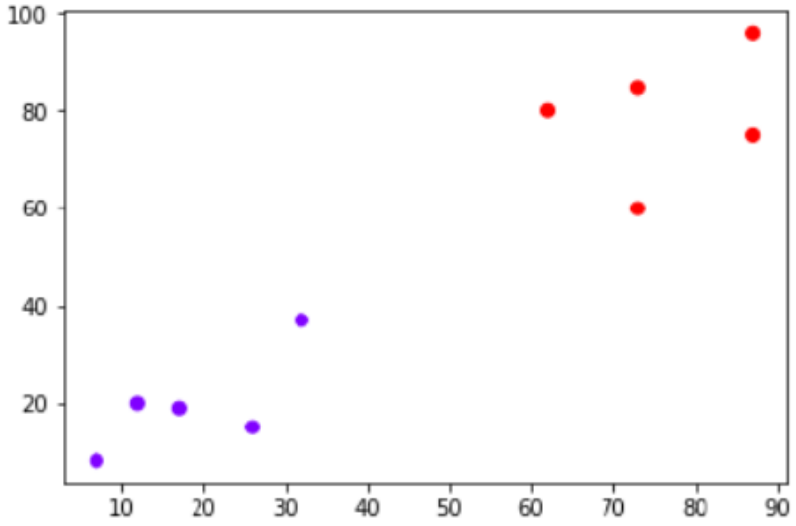
It is one of the simplest and widely used algorithm which depends on it's k value(Neighbors) and finds it's applications in many industries like finance industry, healthcare industry etc.

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well .

Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

It is used for both classification and regression. It predicts a target variable using one or multiple independent variables. kNN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a good suite category by using K-NN algorithm.



A mechanism that is based on the concept of nearest neighbor and where k is some constant represented by a certain number in a particular context, with the algorithm embodying certain useful features such as the use of input to predict output data points, has an application to problems of various nature, focuses on feature similarity so as to classify data, handle realistic data without making any assumptions, can be effectively used in classification problems; and that R programming provides a robust mechanism for its implementation is known as KNN algorithm in R programming.

KNN algorithm machine learning, in this tutorial we are going to explain classification and regression problems.
Machine learning is a subset of artificial intelligence which provides machines the ability to learn automatically and improve from previous experience without being explicitly programmed.
The major part of machine learning is data. Feed the machine with data and make a model and predict. Feed with more data and the model becomes more accurate accordingly.

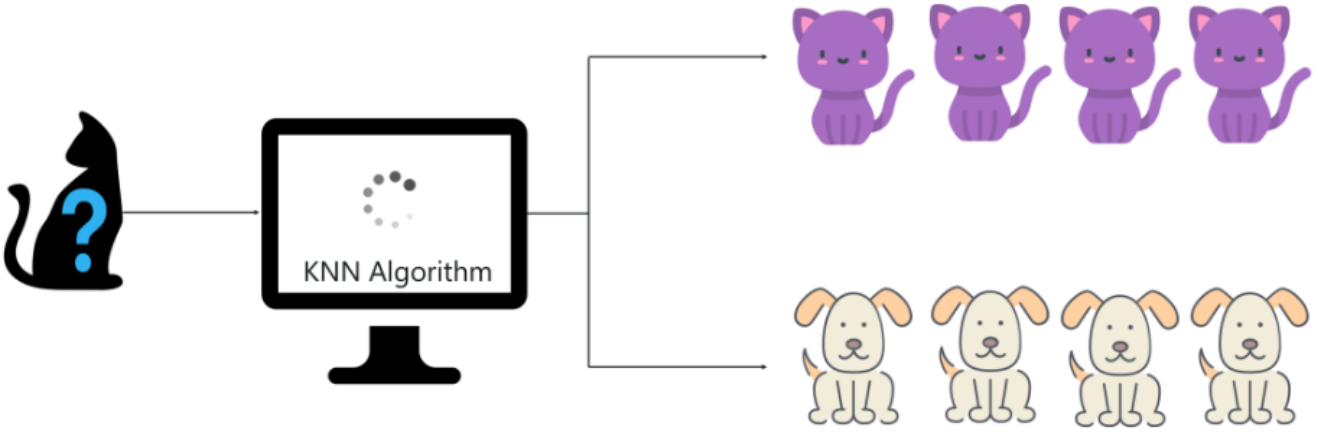
The huge amount of data that we're generating every day, has led to an increase of the need for advanced Machine Learning Algorithms. One such well-performed algorithm is the K Nearest Neighbour algorithm.
In this blog on KNN Algorithm In R, we will understand what is KNN algorithm in Machine Learning and its unique features including the pros and cons, how the KNN algorithm works, an essay example of it, and finally moving to its implementation of KNN using the R Language.
It is quite essential to know Machine Learning basics. Here's a brief introductory section on what is Machine Learning and its types.

KNN which stands for **K Nearest Neighbor** is a Supervised Machine Learning algorithm that classifies a new data point into the target class, counting on the features of its neighboring data points.
Let's attempt to understand the KNN algorithm with an essay example. Let's say we want a machine to distinguish between the sentiment of tweets posted by various users. To do this we must input a dataset of users' sentiment(comments). And now, we have to train our model to detect the sentiments based on certain features. For example, features such as labeled tweet sentiment i.e., as positive or negative tweets accordingly. If a tweet is positive, it is labeled as 1 and if negative, then labeled as 0.

In order to make understand how KNN algorithm works, let's consider the following scenario:
In the image, we have two classes of data, namely class A and Class B representing squares and triangles respectively.

The problem statement is to assign the new input data point to one of the two classes by using the KNN algorithm
The first step in the KNN algorithm is to define the value of 'K' which stands for the number of Nearest Neighbors.

In this image, let's consider 'K' = 3 which means that the algorithm will consider the three neighbors that are the closest to the new data point. The closeness between the data points is calculated either by using measures such as Euclidean or Manhattan distance. Now, at 'K' = 3, two squares and 1 triangle are seen to be the nearest neighbors. So, to classify the new data point based on 'K' = 3, it would be assigned to Class A (squares).



What is KNN Algorithm? – KNN Algorithm In R – Edureka

After studying the dataset during the training phase, when a new image is given to the model, the KNN algorithm will classify it into either cats or dogs depending on the similarity in their features. So if the new image has pointy ears, it will classify that image as a cat because it is similar to the cat images. In this manner, the KNN algorithm classifies data points based on how similar they are to their neighboring data points.

SDD :

The SSD object detection composes of 2 parts:

1. Extract feature maps, and SSD uses VGG16 to extract feature maps. Then it detects objects using the Conv4_3 layer. For illustration, we draw the Conv4_3 to be 8 × 8 spatially (it should be 38 × 38). For each cell in the image(also called location), it makes 4 object predictions.

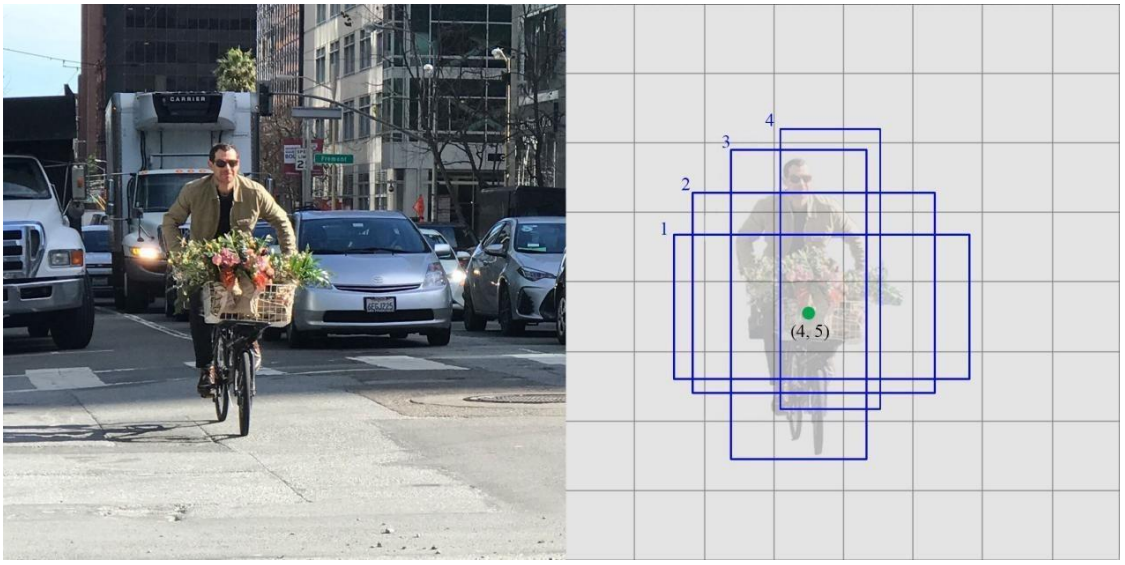


Figure 3.1.6

Each prediction composes of a boundary box and 21 scores for each class (one extra class for no object), and we pick the highest score as the class for the bounded object. Conv4_3 makes total of 38× 38 × 4 predictions: four predictions per cell regardless of the depth of featuremaps. A expected,many predictions contain no object. SSD reserves a class “0” to indicate

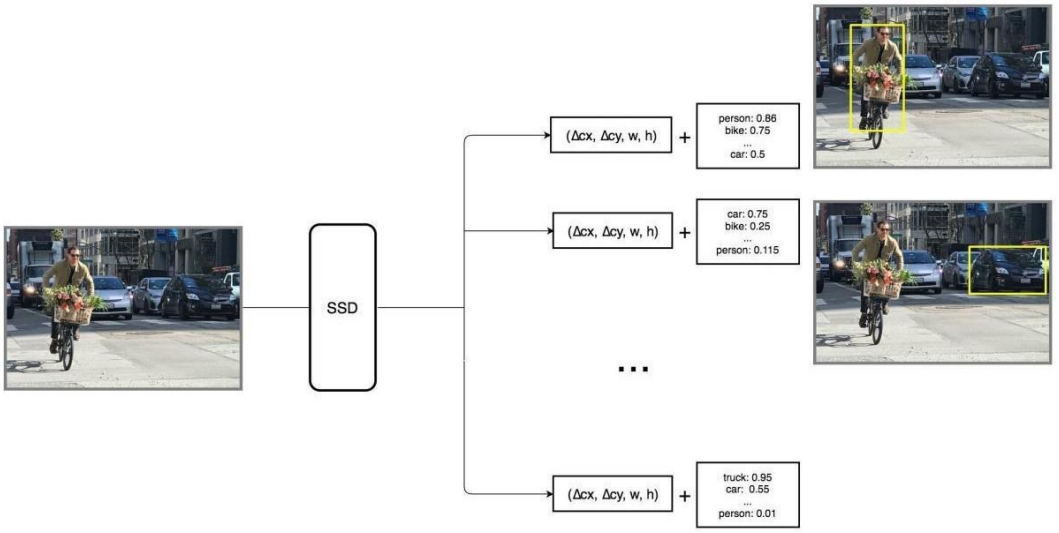


Figure 3.1.6

SSD does not use the delegated region proposal network. Instead, it resolves to a very simple method. It computes both the location and class scores using small convolution filters. After extraction the feature maps, SSD applies 3 × 3 convolution filters for each cell to make predictions. (These filters compute the results just like the regular CNN filters.) Each filter gives outputs as 25 channels: 21 scores for each class plus one boundary box.

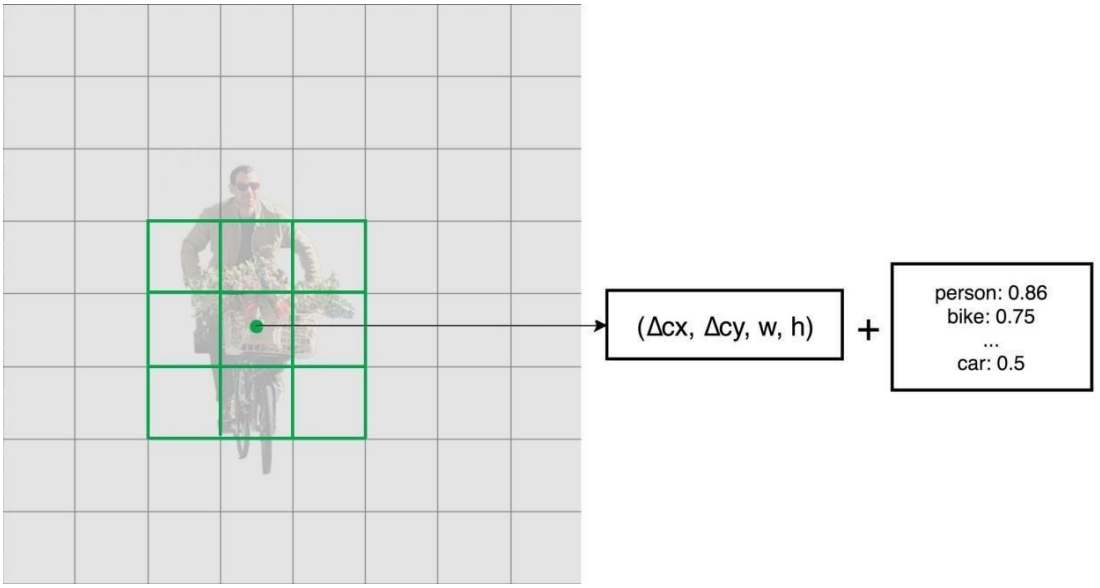


Figure 3.1.6

Beginning, we describe the SSD detects objects from a single layer. Actually, it uses multiple layers (multi-scale feature maps) for the detecting objects independently. As CNN reduces the spatial dimension gradually, the resolution of the feature maps also decrease. SSD uses lower resolution layers for the detect larger-scale objects.

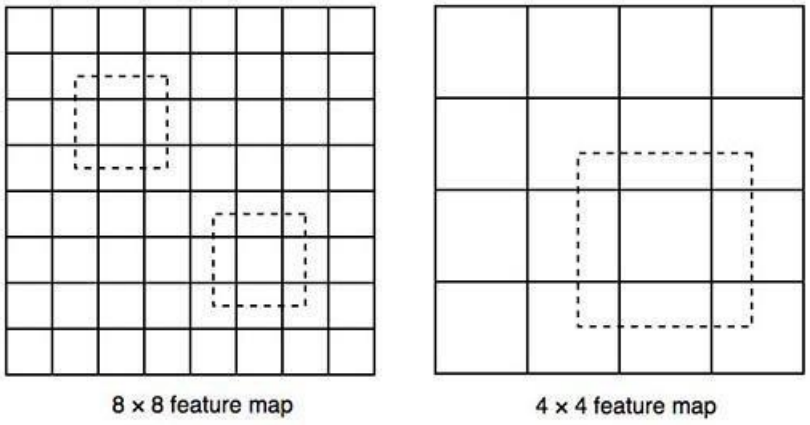


Figure 3.1.6

SSD adds 6 more auxiliary convolution layers to image after VGG16. Five of these layers will be added for object detection. In which three of those layers, we make 6 predictions instead of 4. In total,SSD makes 8732 predictions using 6 convolution layers.

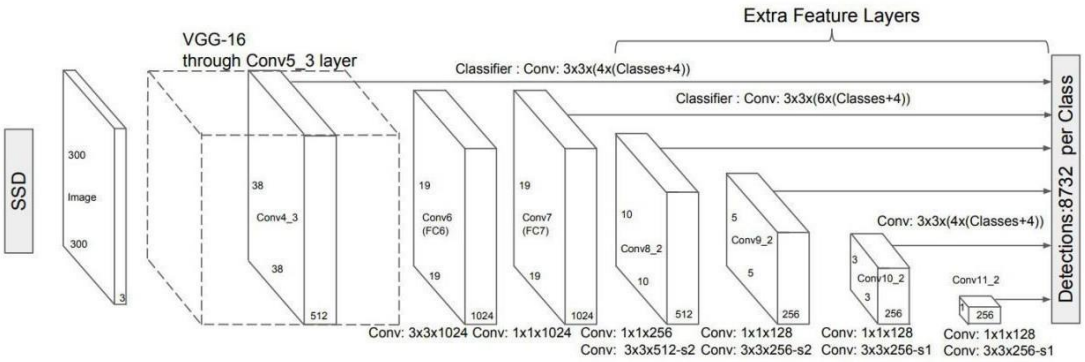


Figure 3.1.6

Multi-scale feature maps enhance accuracy. The accuracy with different number of feature map layersis used for object detection.

Comparison of Existing vs Proposed system

Existing web-based object detection systems generally use pre-trained models and APIs provided by popular computer vision libraries such as TensorFlow, PyTorch, or OpenCV. These models have been trained on large datasets and can detect a wide range of objects with high accuracy.

In our application a real-time object detection system using the TensorFlow.js library, which allows for client-side processing of deep learning models. The system captures video from a user's camera, processes each frame using an object detection model, and draws bounding boxes around detected objects with their corresponding labels and confidence scores.

Compared to existing web-based object detection systems, the in our application has some advantages and disadvantages:

Advantages:

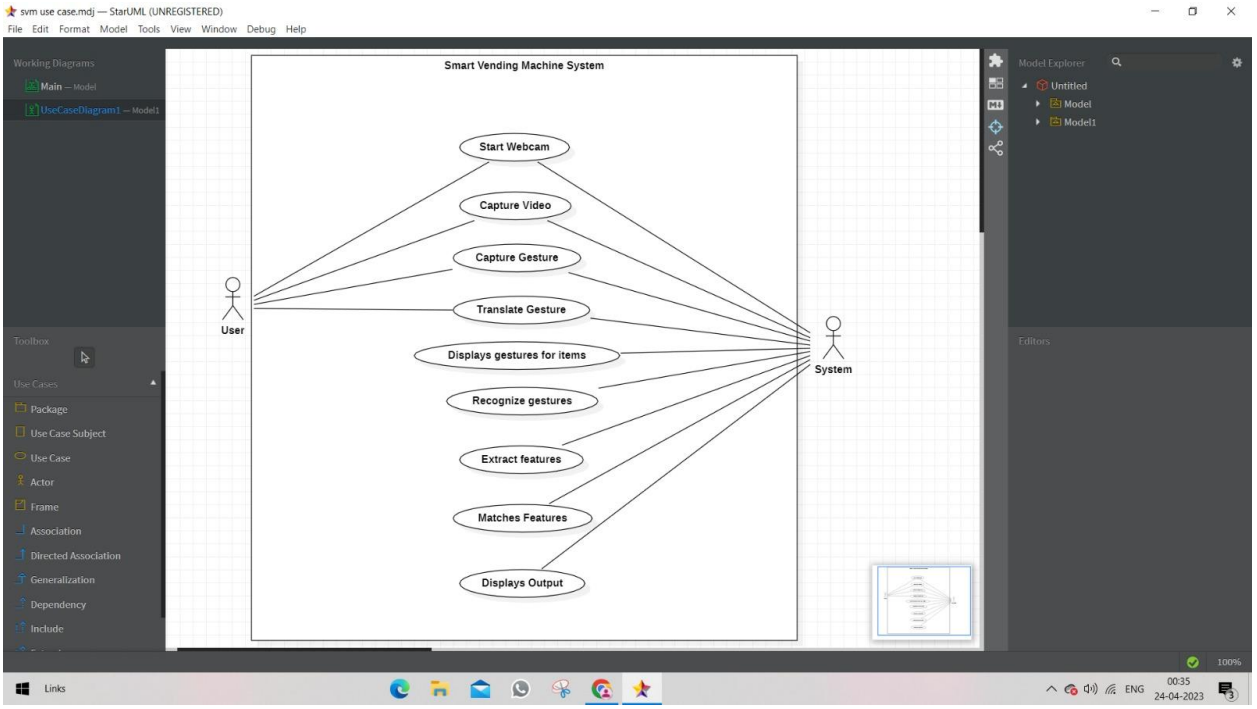
- Real-time object detection: The system processes each frame in real-time, providing instantaneous feedback to the user.
- Client-side processing: The system runs entirely on the user's browser, eliminating the need for server-side processing and potentially reducing latency.
- Customization: The code can be easily customized and modified to use different object detection models or incorporate additional features.

Disadvantages:

- Limited accuracy: The object detection model used in the code may not be as accurate as more complex models used in existing systems.
- Limited object detection capabilities: The system may not be able to detect all types of objects, especially those that are not included in the training dataset of the object detection model.
- Limited scalability: The system may not be able to handle large numbers of users or heavy traffic, as all processing is done on the client-side.

Overall, the above code provides a simple and customizable real-time object detection system, but may not be as accurate or scalable as more complex existing systems that use pre-trained models and server-side processing.

Use Case Diagram : -



Chapter 4. Proposes

1. Based on the potential applications of real-time object detection in web applications, we propose several possible directions for future work:
2. Improving the performance of the model: The current code uses a pre-trained model for object detection, which may not be optimized for the specific application. Future work could involve training a custom model or fine-tuning the pre-trained model to improve accuracy and reduce computation time.

3. Developing more advanced features: While the current code provides basic object detection capabilities, future work could include developing more advanced features such as tracking objects over time, identifying specific atributes of objects, and recognizing complex scenes.

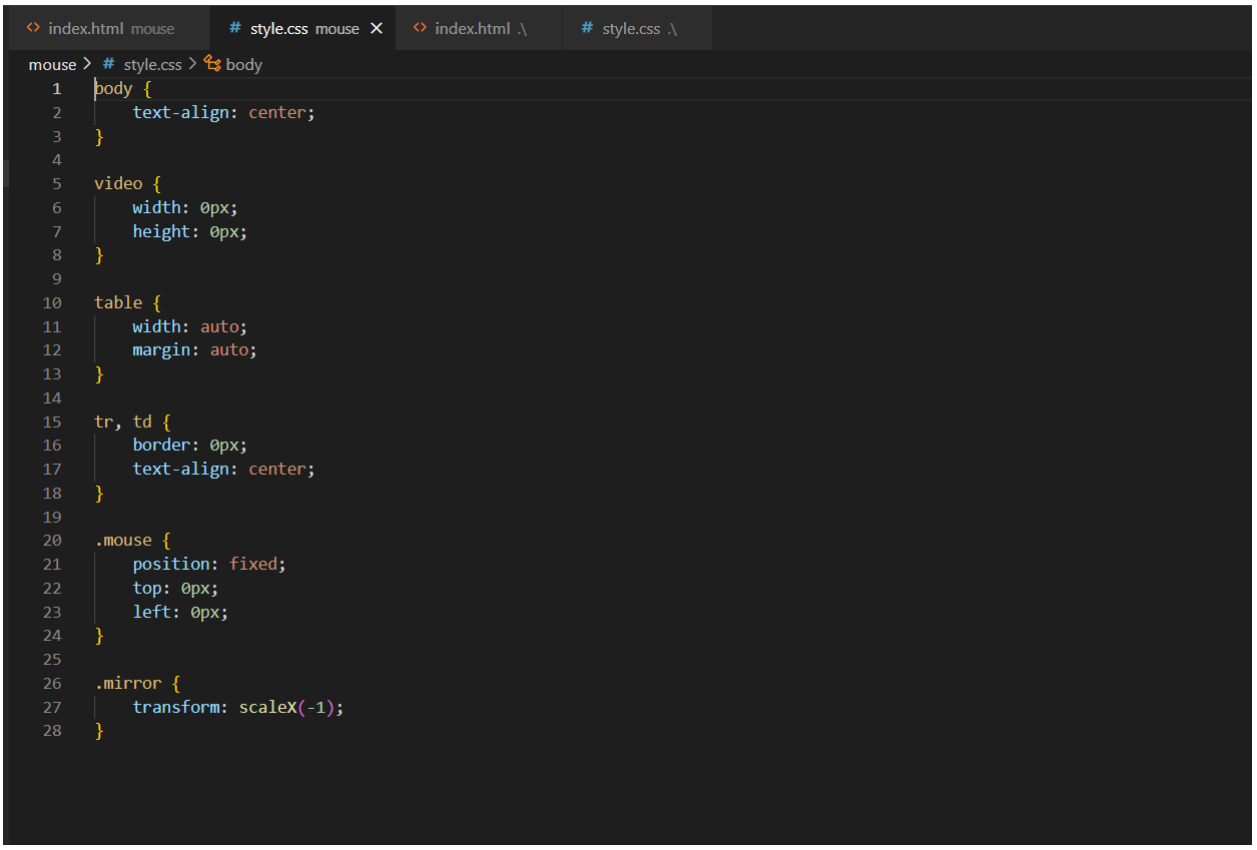
4. Integrating with other technologies: Real-time object detection can be integrated with other technologies such as augmented reality and virtual reality to create more immersive and interactive experiences. Future work could explore how real-time object detection can be integrated with these technologies to create novel applications.

5. Enhancing security: Real-time object detection can be used to enhance security in various fields, such as identifying potential threats in public spaces or detecting fraudulent activities in e-commerce. Future work could focus on developing applications that utilize real-time object detection for security purposes.

Overall, the potential of real-time object detection in web applications is vast, and future work in this field can lead to exciting new applications and opportunities.

Chapter 5.
Implementation

```
index.html mouse X | # style.css mouse | index.html .\ | # style.css .\
mouse > index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <title>Hand mouse control</title>
6    <meta charset="utf-8">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
9    <link rel="stylesheet" href="style.css">
10   <script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
11 </head>
12
13 <body>
14   <h2 id="loadingText">Loading...</h2>
15   <!-- video with size of 0px because of chrome -->
16   <video playsinline autoplay muted controls="true" id="video"></video>
17   <br><br>
18   <canvas id="c1" class="mirror"></canvas>
19   <canvas id="mouse" class="mouse"></canvas>
20   <br><br>
21   <table>
22     <tr>
23       <td>AI:</td>
24       <td>
25         <div class="switch">
26           <label>
27             Off
28             <input type="checkbox" id="ai" disabled>
29             <span class="lever"></span>
30             On
31           </label>
32         </div>
33       </td>
34     </tr>
35     <tr>
36       <td>Border:</td>
37       <td>
```

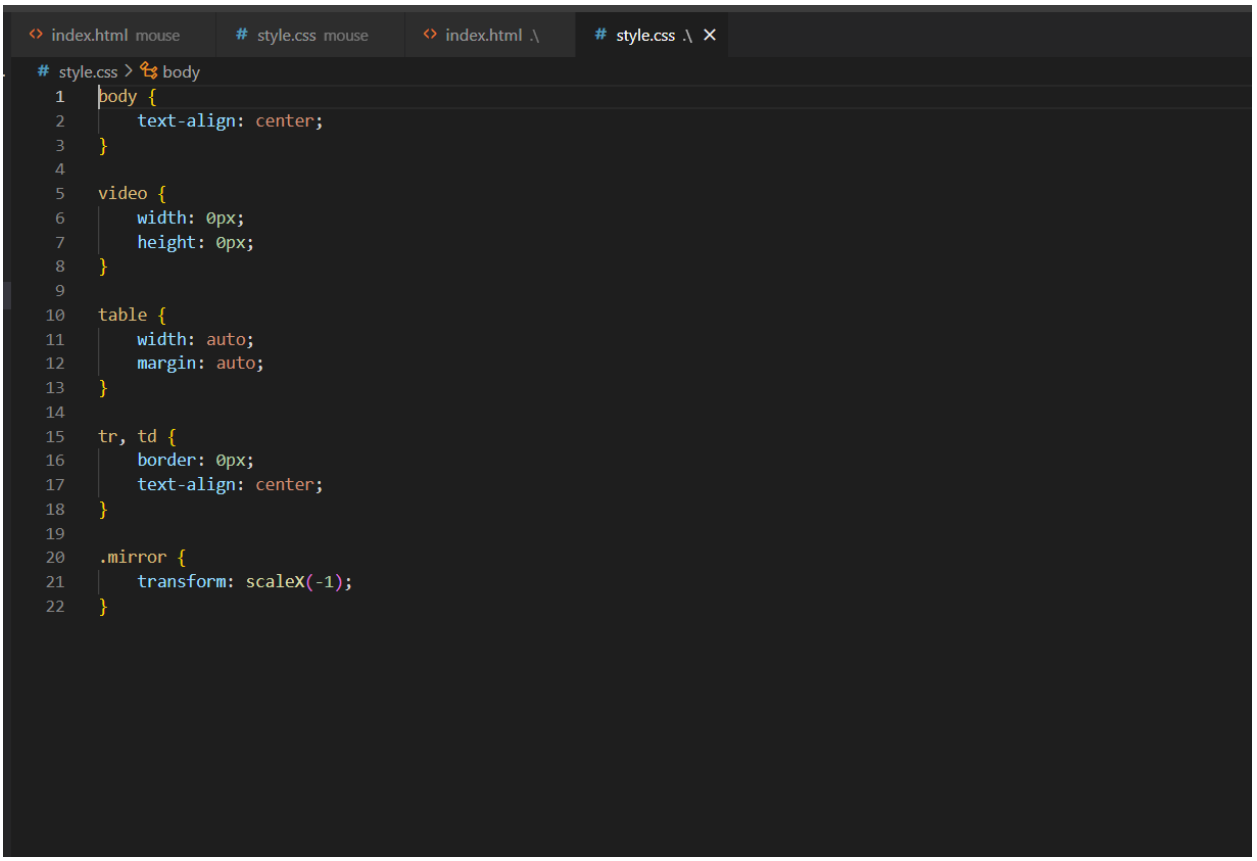

index.html mouse
style.css mouse
index.html .\
style.css .\
mouse > style.css > body
1 body {
2 text-align: center;
3 }
4
5 video {
6 width: 0px;
7 height: 0px;
8 }
9
10 table {
11 width: auto;
12 margin: auto;
13 }
14
15 tr, td {
16 border: 0px;
17 text-align: center;
18 }
19
20 .mouse {
21 position: fixed;
22 top: 0px;
23 left: 0px;
24 }
25
26 .mirror {
27 transform: scaleX(-1);
28 }

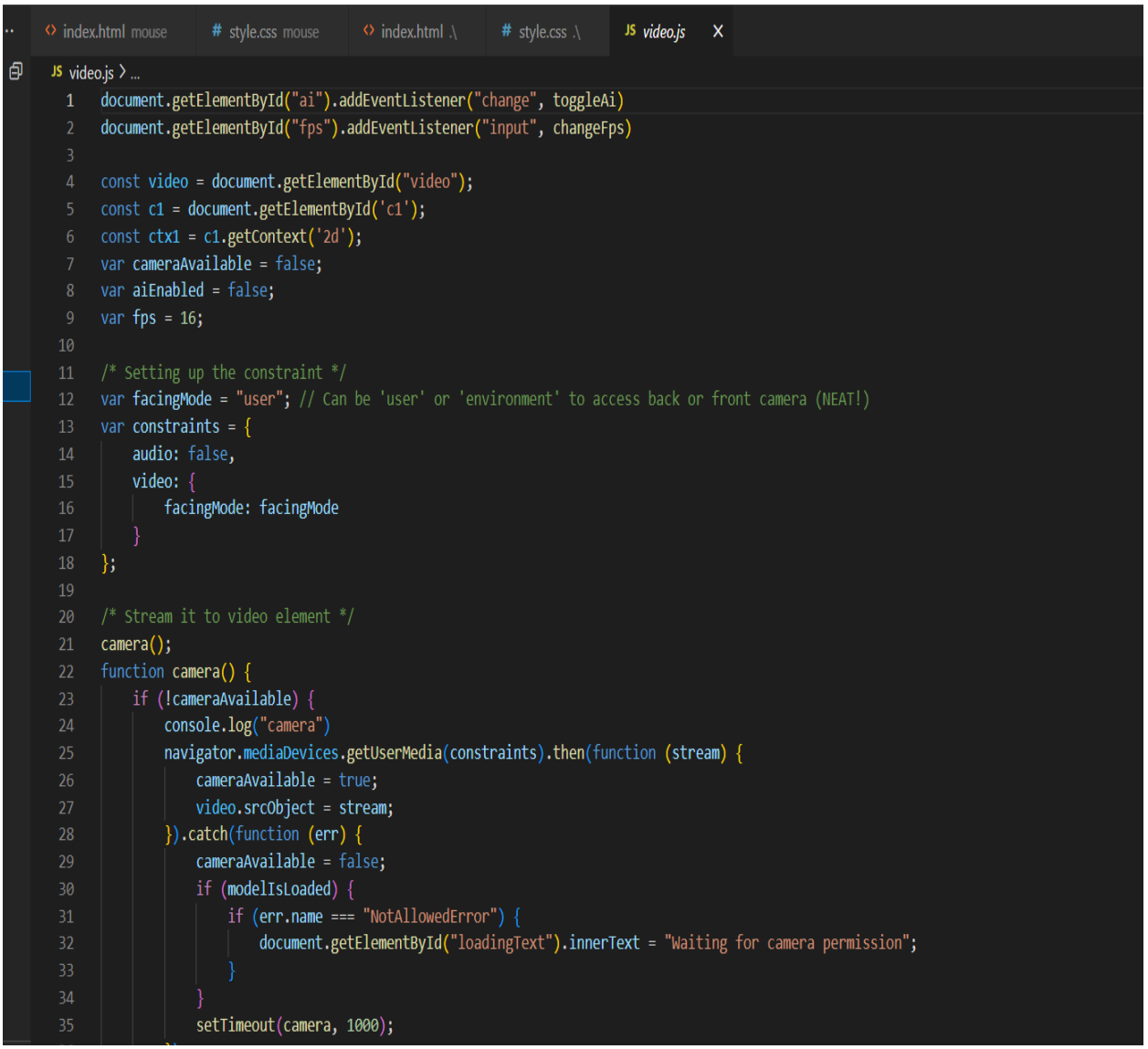
index.html > ...

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <title>AI hand detection</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
  <link rel="stylesheet" href="style.css">
  <script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
</head>

<body>
  <h2 id="loadingText">Loading...</h2>
  <!-- video with size of 0px because of chrome -->
  <video playsinline autoplay muted controls="true" id="video"></video>
  <br><br>
  <canvas id="c1" class="mirror"></canvas>
  <br><br>
  <table>
    <tr>
      <td>AI:</td>
      <td>
        <div class="switch">
          <label>
            Off
            <input type="checkbox" id="ai" disabled>
            <span class="lever"></span>
            On
          </label>
        </div>
      </td>
    </tr>
    <tr>
      <td>FPS:</td>
      <td>
        <p class="range-field">
```


index.html mouse
style.css mouse
index.html .\
style.css .\
style.css > body
1 body {
2 text-align: center;
3 }
4
5 video {
6 width: 0px;
7 height: 0px;
8 }
9
10 table {
11 width: auto;
12 margin: auto;
13 }
14
15 tr, td {
16 border: 0px;
17 text-align: center;
18 }
19
20 .mirror {
21 transform: scaleX(-1);
22 }


index.html mouse
style.css mouse
index.html .\
style.css .\
video.js
video.js > ...
document.getElementById("ai").addEventListener("change", toggleAi)
document.getElementById("fps").addEventListener("input", changeFps)
const video = document.getElementById("video");
const c1 = document.getElementById('c1');
const ctx1 = c1.getContext('2d');
var cameraAvailable = false;
var aiEnabled = false;
var fps = 16;
/* Setting up the constraint */
var facingMode = "user"; // Can be 'user' or 'environment' to access back or front camera (NEAT!)
var constraints = {
audio: false,
video: {
facingMode: facingMode
}
};
/* Stream it to video element */
camera();
function camera() {
if (!cameraAvailable) {
console.log("camera")
navigator.mediaDevices.getUserMedia(constraints).then(function (stream) {
cameraAvailable = true;
video.srcObject = stream;
}).catch(function (err) {
cameraAvailable = false;
if (modelIsLoaded) {
if (err.name === "NotAllowedError") {
document.getElementById("loadingText").innerText = "Waiting for camera permission";
}
}
setTimeout(camera, 1000);

This code is a JavaScript implementation of real-time object detection using a pre-trained object detection model. The code uses the getUserMedia() function to access the camera stream and displays the video on a canvas element. The canvas element is then passed through the object detection model to detect objects in real-time.

The code starts by adding event listeners to the toggle button for enabling and disabling the object detection feature and to the input range element for changing the frames per second (fps) rate. It then initializes variables for the video element, canvas element, and the camera availability status. It also sets up the camera constraint to access the back or front camera based on the "environment" or "user" mode respectively.

The camera() function is then called to initiate the camera stream and check for any errors. If the camera stream is available, the video element is assigned the stream object. If not, the function waits for a second and tries again.

The timerCallback() function is called to constantly check if the model is loaded and if the camera stream is available. If both conditions are true, the function sets the resolution of the canvas element based on the screen size and the video element dimensions. It then draws the video element onto the canvas element and calls the ai() function if the object detection feature is enabled.

The setResolution() function sets the canvas element size based on the aspect ratio of the video element and the screen size.

The toggleAi() function is called when the user clicks on the toggle button to enable or disable the object detection feature. It sets the value of the aiEnabled variable accordingly.

The changeFps() function is called when the user changes the fps rate using the input range element. It sets the value of the fps variable based on the value of the input range element.

The ai() function passes the canvas element through the object detection model using the detect() method. The results are logged in the console, and bounding boxes are drawn around the detected objects on the canvas element with their labels and confidence scores.

This code is a JavaScript implementation of a real-time object detection application using TensorFlow.js and an object detection model trained on the COCO dataset. The application streams video from the user's camera and performs object detection on each frame using the TensorFlow.js Object Detection API.

The code first sets up the constraints for accessing the user's camera and then sets up a callback function to continually check if the model is loaded and the camera is available. If both conditions are true, the application begins processing each frame of the video using theobject detection model.

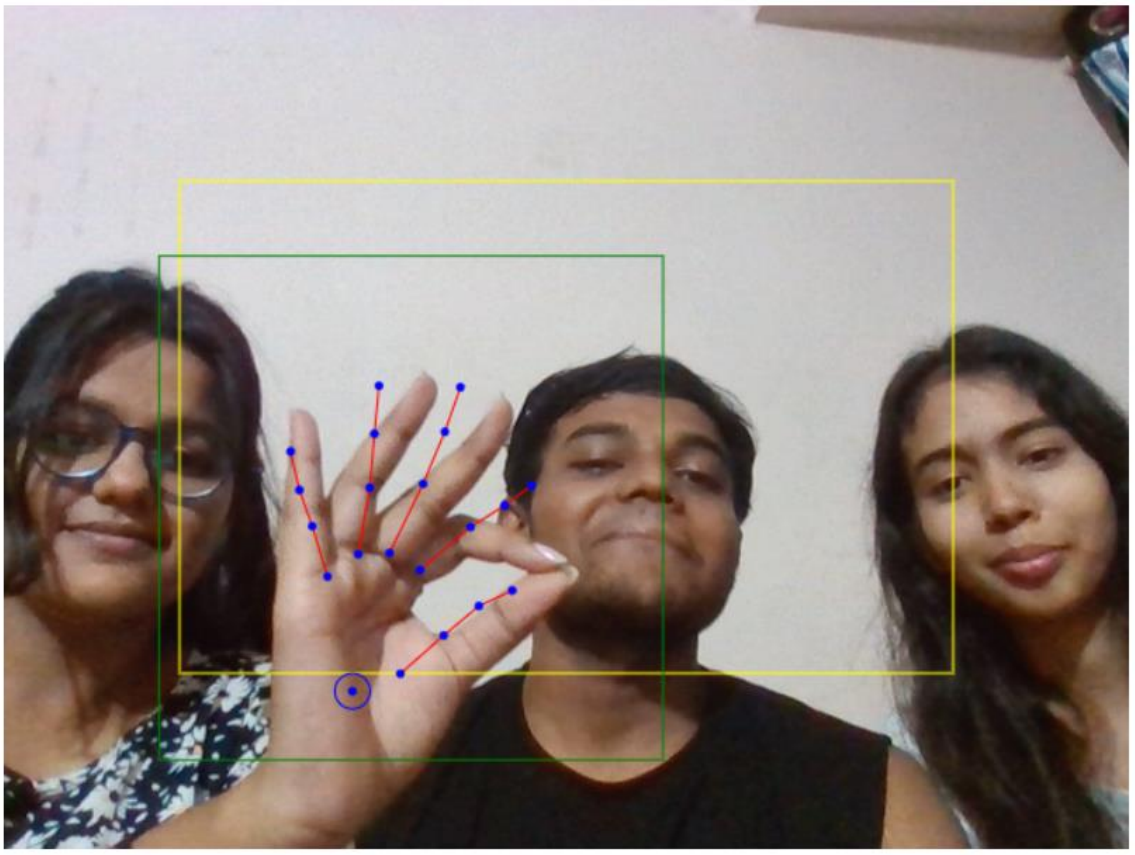
The code uses the canvas element to draw the video feed and the detected objects on the screen. It also provides options to toggle the object detection feature on and off and to adjustthe frame rate of the video stream.

The ai() function performs the object detection using the TensorFlow.js Object Detection API. It takes the canvas element as input and outputs the bounding boxes of the detected objects. The code then uses the bounding box information to draw the labels and boxes around eachdetected object on the canvas element.

Chapter 6. Result

As this code is a real-time object detection application, the result would depend on the objects present in the camera's view. The application uses the COCO-SSD model for object detection, which is trained on the Common Objects in Context (COCO) dataset. The model is capable of detecting 80 different object classes, including people, animals, vehicles, and various household and everyday objects.

Once the objects are detected, their labels and confidence scores are displayed on the video feed using red bounding boxes and text labels. The performance of



AI: Off On

Border:

Link of Our Website, Which is hosted on Github (Live Demo) : -

Click Here : - https://woody.pizza/tensorflow/hand-detection/mouse

Chapter 7.
Conclusion

The model of the vending machine successfully detects the coin and dispenses liquid output. It is also proved to be customer friendly. The system developed in this research work can be used as a base for future development.

First is the coin detection section. When the coin is inserted it comes across the CS-616 coin acceptor and IR sensor which detects the coin using thus recognizing that it is not a piece of paper or other metal. The size of Rs.10 coin is identified by using two pairs of IR sensors one at the beginning of insertion and the other at the end equal to the currency length. (Size 137*63mm of Rs.10)
Second section is the placing of order.First the LCD displays to insert the coin.Then the customer inserts the coin.When 10 rupee coin is detected, the LCD displays select your drink else if not detected displays wrong coin inserted..
Last section is the dispensing of required beverage. When we select the button, the relay gets activated and liquid output is dispensed into the cup. The time is already adjusted according to the size of cup so that liquid does not flows out.

Chapter 8. References

1. Agarwal, S., Awan, A., and Roth, D. (2004). Learning to Vending Machine in images viaasparse,part-based representation. IEEE Trans. Patern Anal. Mach. Intell. 26,1475–1490.doi:10.1109/TPAMI.2004.108
2. Alexe, B., Deselaers, T., and Ferrari, V. (2010). "What is an object?," in ComputerVisionandPaternRecognition (CVPR), 2010 IEEE Conference on (San Francisco,CA: IEEE), 73–80.doi:10.1109/CVPR.2010.5540226
3. Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. Int. J.Comput. Vis.1,333–356. doi:10.1007/BF00133571
4. Andreopoulos, A., and Tsotsos, J. K. (2013). 50 years of object recognition: direc- tionsforward.Comput. Vis. Image Underst. 117, 827–891.doi:10.1016/j.cviu.2013.04.005
5. Azizpour, H., and Laptev, I. (2012). "Object detection using strongly-supervisedeformablepartmodels," in Computer Vision-ECCV 2012 (Florence: Springer),836–849.
6. Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detectionandpaternrecognition. IEEE Trans. Patern Anal. Mach. Intell. 35, 490– 503.doi:10.1109/TPAMI.2012.106
7. Azzopardi, G., and Petkov, N. (2014). Ventral-stream-like shape representation:frompixel intensityvalues to trainable object-selective cosfire models. Front.Comput. Neurosci. 8:80.doi:10.3389/fncom.2014.00080
8. Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). "Fast classification using sparsedecisiondags,"in Proceedings of the 29th International Conference on MachineLearning (ICML-12), ICML'12, edsJ. Langford and J. Pineau (New York, NY:Omnipress), 951–958.
9. Bengio, Y. (2012). "Deep learning of representations for unsupervised and transferlearning,"inICML Unsupervised and Transfer Learning, Volume 27 of JMLRProceedings, eds I. Guyon, G. Dror,V. Lemaire, G. W. Taylor, and D. L. Silver(Bellevue: JMLR.Org), 17–36.
10. Bourdev, L. D., Maji, S., Brox, T., and Malik, J. (2010). "Detecting peopleusing mutuallyconsistentposelet activations," in Computer Vision.