PROJECT TITLE : OBJECT DETECTION USING TINY ML

SUBJECT : COMPUTER HARDWARE AND SOFTWARE

SUBJECT CODE : COCSC19

GROUP MEMBERS :

1) 2021UCS1663 RAJ KUMAR

2) 2021UCS1670 ROSHAN JEET KUMAR

3) 2021UCS1697 ANKIT RAJ

TABLE OF CONTENTS :

1. INTRODUCTION TO TINY ML :

Tiny machine learning is broadly defined as a fast-growing field of machine learning technologies and applications including hardware, algorithms, and software capable of performing on-device sensor data analytics at extremely low power, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices.

These are the following benefits of tiny ml.

1. **Latency**: The data does not need to be transferred to a server for inference because the model operates on edge devices. Data transfers typically take time, which causes a slight delay. Removing this requirement decreases latency.

2. **Energy savings**: Microcontrollers need a very small amount of power, which enables them to operate for long periods without needing to be charged. On top of that, extensive server infrastructure is not required as no information transfer occurs: the result is energy, resource, and cost savings.

3. Reduced bandwidth: Little to no internet connectivity is required for inference. There are on-device sensors that capture data and process it on the device. This means there is no raw sensor data constantly being delivered to the server.

4. Data privacy: Your data is not kept on servers because the model runs on the edge. No transfer of information to servers increases the guarantee of data privacy.

2. INTRODUCTION TO YOLOV3 :

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds or images. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect objects located in an image.

As typical for object detectors, the features learned by the convolutional layers are passed onto a classifier which makes the detection prediction. In YOLO, the prediction is based on a convolutional layer that uses 1×1 convolutions. YOLO stands for “you only look once” because its prediction uses 1×1 convolutions. This means that the size of the prediction map is exactly the size of the feature map before it. This efficient use of 1×1 convolutions contributes to streamlining the prediction process, allowing the fully connected layer to leverage the compact representation of features before making detection predictions.

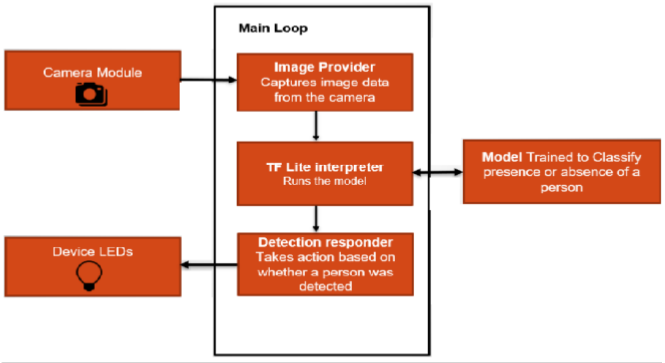
YOLO is a Convolutional Neural Network (CNN), a type of deep neural network, for performing object detection in real time. CNNs are classifier-based systems that process input images as structured arrays of data and recognize patterns between them. YOLO has the advantage of being much faster than other networks and still maintains accuracy.

3. ABSTRACT : Object detection is a challenging problem in computer vision. It has been widely applied in defense military, transportation, industry, etc. To facilitate in-depth understanding of object detection, we comprehensively review the existing object detection methods based on deep learning from four aspects, including multi-scale feature learning, data augmentation, training strategy, context-based detection. Tiny ML is a fast-growing multidisciplinary field at the inter section of machine learning, hardware, and software, that focuses on enabling deep learning algorithms on embedded (microcontroller powered) devices operating at extremely low power range (mW range and below). Tiny ML addresses the challenges in designing power efficient, compact deep neural network models, supporting software frame work, and embedded hardware that will enable a wide range of customized, ubiquitous inference applications on battery-operated, resource-constrained devices.

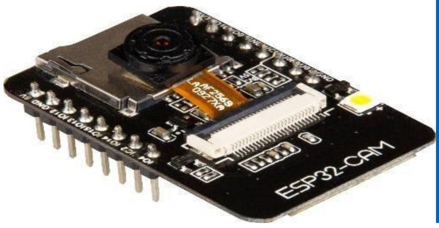
4. INTRODUCTION AND OVERVIEW OF PROJECT :

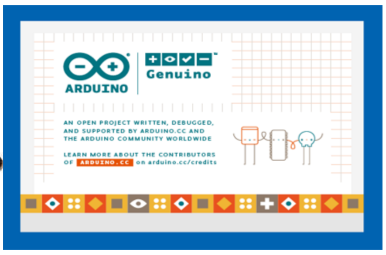
Object detection technic is to detect the activity of a person which can be restricted or unauthorized person can be stopped by using this system this will achieve good security for critical information or products. Object detection is a computer vision technique that allows us to identify and locate objects in an image. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Reliability is the main attribute for safe operation in any modern technological system

5. SYSTEM ARCHITECTURE :


Main Loop
Camera Module
Image Provider Captures image data from the camera
TF Lite interpreter Runs the model
Model Trained to Classify presence or absence of a person
Device LEDs
Detection responder Takes action based on whether a person was detected

COMPONENTS USED: Esp 32 camera module , Arduino ide




ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

6. APPLICATIONS :

These are the following applications of object detection.

1. Healthcare: In healthcare, specifically in the radiology sector, object detection can be used to identify and localize tumors and other abnormalities in medical images such as MRIs, CT scans, x-rays, etc. This can help doctors and radiologists make more accurate diagnoses and develop more effective treatment plans.

2. Retail: In retail, object detection can be used to optimize inventory management and store security. Retailers can use object detection to track inventory levels by scanning shelves at stores and identifying when products are low in stock.

4. Manufacturing : Manufacturing products like cars require the assembly of thousands of components, and keeping track of them can be repetitive and error-prone if done manually. Object detection can be used to scan products on assembly lines. This enables improved quality control and ensures that products are assembled correctly.

5. Agriculture: Many agricultural companies are now adopting digital solutions which leverage object detection. Object detection can be used in areas such as:Crop monitoring,Yield estimation, pest detection. This can help farmers take corrective measures before the issue ruins the whole crop. For instance, For instance, a recent study presented an object detection algorithm to identify and monitor tomato plants infected with a bacterial disease called speck.

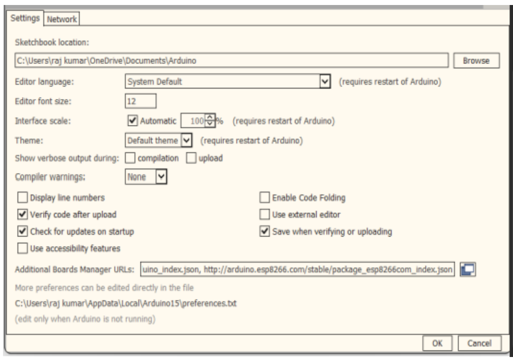
7. WORKFLOW AND DEPLOYMENT:

1. Installation of Arduino ide : we have installed Arduino version 1.8.15 in our system and finished its set up. Here is the link to download Arduino version 1.8.15.

https://r.search.yahoo.com/_ylt=AwrKEZTpjLdlWhUU2EW7HAx.;_ylu=Y29sbwNzZzMEcG9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1706556778/RO=10/RU=https%3a%2f%2ffilehippo.com%2fdownload_arduino-ide%2f1.8.15%2f/RK=2/RS=nAGH_EPGcqHf5vCHCeqYctr6ijs-

2. Installation of esp32 on Arduino kit :

- Start Arduino and open the Preferences window and add the link given in the additional board manager urls:
  https://dl.espressif.com/dl/package_esp32_index.json,http://dan.drown.org/stm32duino/package_STM32duino_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json



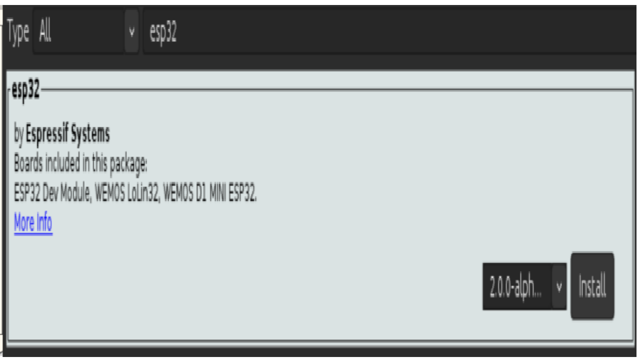
Settings | Network

Sketchbook location: C:\Users\raj kumar\OneDrive\Documents\Arduino [Browse]
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None

☐ Display line numbers
[✓ Verify code after upload]
[✓ Check for updates on startup]
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
[✓ Save when verifying or uploading]

Additional Boards Manager URLs: uino_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json
More preferences can be edited directly in the file
C:\Users\raj kumar\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

OK | Cancel



Type All | esp32

esp32
by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

2.0.0-alph... | Install

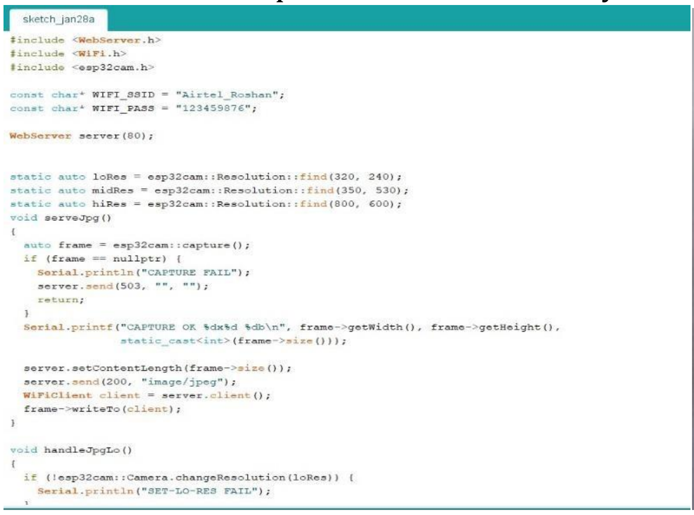
Now esp 32 is successfully installed in the Arduino kit. We can also check whether it is installed or not by typing esp32 in search box of board manager.

3. Adding esp32 zip library : After installing esp32 in the Arduino kit. We have to include esp32 zip file so that we can connect esp32 camera module with our system for detection of objects. Here is the link to download esp32 main zip file.https://github.com/yoursunny/esp32cam. To include zip library go to sketch bar in Arduino kit and then include library and then add .zip library.

4. Connect esp32 camera with laptop : Connect the ESP32 board to the PC using the USB cable. If device driver does not install automatically, identify USB-to-UART bridge on your ESP32 board (or external converter dongle). Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps:

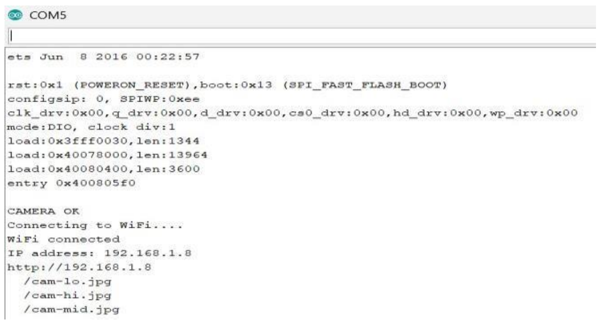
Select your Board in tools > board menu , choose esp wrover kit, Select the Port, set baud rate to 115200, upload sketch. Then click on reboot button which is done by clicking on RST button and IO0 button simultaneously. After that esp32 camera module is successfully connected to the laptop.

And then click on share monitor a screen will open in which camera ok is written which indicates that the esp32 camera is now ready to detect objects.



sketch_jan28a

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

const char* WIFI_SSID = "Airtel_Roshan";
const char* WIFI_PASS = "123459876";

WebServer server(80);


static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
```



```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0

CAMERA OK
Connecting to WiFi....
WiFi connected
IP address: 192.168.1.8
http://192.168.1.8
  /cam-lo.jpg
  /cam-hi.jpg
  /cam-mid.jpg
```

5. Dataset collection : We have collected various objects which we are going to detect. This dataset is saved in file coco.names. Here is the link for the file.
https://1drv.ms/u/s!AjiCykDqYnWnjWearf82DPMb7T5L?e=3KI8Ic

5. Python code:

We have made .py file in which implementation for object detection in has been done. Here is the colab link for the python file.

https://colab.research.google.com/drive/16fDgo9b6XxLlS4--lqbKuj49TMsZBce5?usp=sharing

This Python code is a computer vision application that performs object detection using a YOLO (You Only Look Once) v3 model on a live video stream from an ESP32 camera. Here's a breakdown of what the code does:

Imports: cv2: OpenCV library for computer vision tasks, numpy: Library for numerical computations, urllib.request: Library for handling URLs, time: Library for time-related operations.

Setup: stream_url: URL of the ESP32 camera stream, Constants like whT (width and height), confThreshold (confidence threshold), and nmsThreshold (non-maximum suppression threshold), Loading YOLO v3 model configuration (modelConfig) and weights (modelWeights), Reading class names from coco.names file.

findObject Function:

Processes YOLO output to identify objects.

Draws bounding boxes and labels on detected objects based on confidence scores.

Main Loop (while True):

Continuously attempts to open the camera stream until successful.

Reads the video stream frame by frame.

Detects objects in each frame using YOLO v3.

Draws bounding boxes and labels on detected objects.

Displays the processed frame with detected objects.

Listens for the Esc key to exit the application.

cv2.destroyAllWindows():

Closes all OpenCV windows when the application exits.

The key parts of this code are:

Retrieving frames from the camera stream (urllib.request.urlopen()).

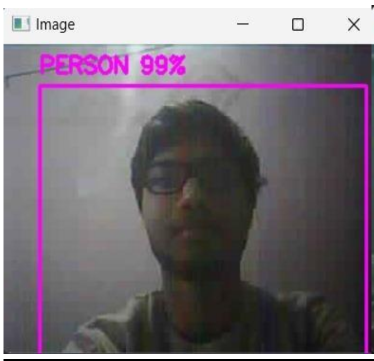
Decoding each frame (cv2.imdecode()).

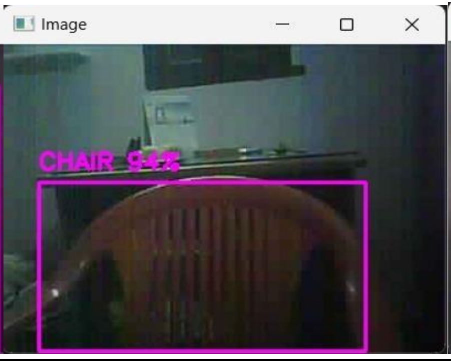
Performing object detection on each frame using YOLO v3.

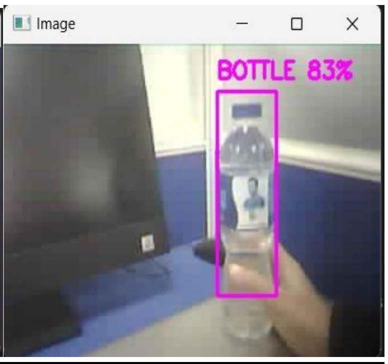
Displaying the processed frame with detected objects (cv2.imshow()).

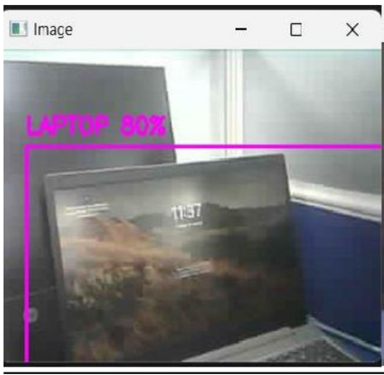
The findObject function is crucial for processing the YOLO output and drawing bounding boxes and labels on detected objects. It iterates through each detected object, applies confidence thresholding, and uses non-maximum suppression to filter out overlapping boxes. Finally, it draws bounding boxes and labels for each detected object on the frame.

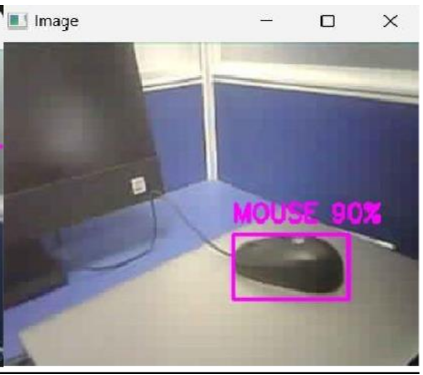
8. RESULT/OUTPUT:











9. CONCLUSION :

object detection using ESP32 is giving our devices a heightened sense of perception. This capability equips them with the ability to visually recognize and understand objects in their surroundings.This advancement facilitates the creation of intelligent applications, particularly in fields like surveillance and

automation.object detection withESP32 enhances our devices, enabling them to "see" and interpret the world around
them, leading to more advanced and responsive interactions with technology.

10. FUTURE WORKS:

Advanced Recognition Algorithms:
Research and implement more advanced recognition algorithms to enhance the accuracy of firearm detection. This could involve exploring deep learning techniques,neural network architectures, or ensemble models for improved precision.

Integration with Cloud Services: Investigate the integration of ESP32-based object detection with cloud services. This could enable more extensive and dynamic databases for object recognition, as well as offloading computational tasks to the cloud for complexscenarios.

11. CHALLENGES :

1.**Small Object Detection**: YOLOv3 may struggle with accurately detecting small objects in an image. The fixed grid size can limit the ability to capture fine-grained details for smaller objects.

2.**Complex Backgrounds:**YOLOv3 may produce false positives or miss detections when faced with complex backgrounds or cluttered scenes,
3.**Smaller sized datasets:** Though deep learning models outperform traditional machine learning approaches by a great margin
4.Occlusion of objects by other objects in the scene can be a challenge. YOLOv3 may have difficulty accurately detecting and localizing partially visible objects.

12. REFERENCES:

[1] T. N. Sainath and C. Parada, ``Convolutional neural networks for small-footprint keyword spotting,'' Google Res., New York, NY, USA, Tech. Rep. 43969, 2015. [Online]. Available: https://storage.googleapis.com/pub-tools-publicpublication-data/pdf/43969.pdf

[2] Q. Wang, A. Terzis and A. Szalay, "A Novel Soil Measuring Wireless Sensor Network", IEEE Transactions on Instrumentation and Measurement, pp. 412–415, 2010.

[3] aliyavari.com/papers/2016_Sensors_SmartFarming/SmartFarm.pdf. Young, The Tech-nical Writers Handbook. MillValley, CA: University Science, 1989.

[4] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object security for constrained rest ful environments(o score)," Work in Progress, 2019.