# Human Detection System Using ESP32 Camera

This paper proposes a framework which can be applied to day-today life situations which make tasks easier. The framework utilizes machine learning techniques, camera and Internet of Things (IoT). The video is captured using esp32cam and is sent through the machine learning algorithm where the detection is done and the feed is sent to the cloud. The proposed framework was implemented using ESP32-CAM, TensorFlow and GoogleAPI.

## INTRODUCTION

According to Arizona State University (ASU), the Tempe campus has over 75000 students enrolled. The ASU Tempe campus features two large libraries, both of which are reportedly overcrowded over the whole semester. People have experienced the difficulty of not being able to find a study space in any of the libraries. Individuals typically waste a lot of time looking for a spot. For example, if a student wishes to study in a peaceful spot, he must physically go there to see if the space is available. This is an issue which needs to resolved, and Internet of Things offers a solution for it.

Human detection system is a noteworthy technology which can be paired up with Internet of Things (IoT) to solve this issue. Furthermore, it can be achieved using various Machine Learning algorithms and computer vision.

## PROPOSED FRAMEWORK AND IMPLEMENTATION

The whole project framework has a simple structure and is divided into 5 main parts:

(1) Pre – Trained COCO dataset, (2) TensorFlow Model, (3) ESP32-CAM with FTDI module, (4) HTML Webpage running on local host. The ESP32-CAM is used with TensorFlow to detect

humans. The TensorFlow libraries are used on COCO-SSD model which is a pre-trained machine learning model.

*1) Pre – Trained COCO dataset.*

Microsoft's COCO dataset is a object recognition, segmentation, and captioning dataset. It is widely used by machine learning and computer vision projects.

*2) TensorFlow Model*

TensorFlow lets developers design data flow diagrams, which are structures that explain how data flows via diagrams or a set of processing nodes. Each node in the graphic represents a mathematical process. Lastly, Python programming language is used to handle TensorFlow.

*3) ESP32-CAM with FTDI module*

AI-Thinker created the ESP32-based Camera Module. The controller is powered by a 32-bit CPU and includes a Wi-Fi + Bluetooth/BLE chip. It has an onboard 520 KB SRAM and an external 4M PSRAM. Its GPIO Pins include UART, SPI, I2C, PWM, ADC, and DAC capabilities [1]. The board lacks a programmer chip. So, any form of USB-to-TTL Module can be used to program this board [2]. The connection between the ESP32-CAM and the FTDI module is shown in figure 1.

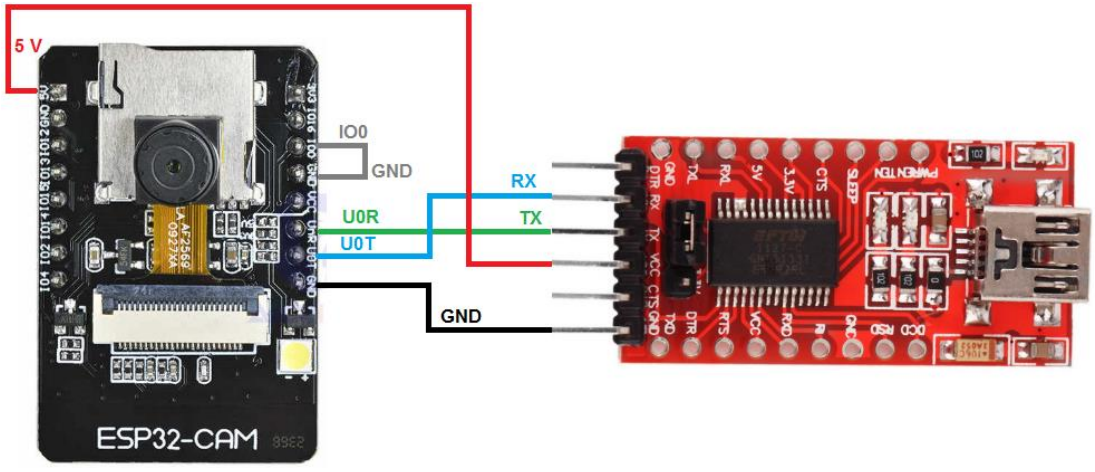


Fig 1: connection between ESP32-CAM and FTDI Module

*4) HTML Webpage*

The entire html webpage is created in the Arduino IDE itself by calling the respective source files of TensorFlow,js which are then used to identify humans. Along with TensorFlow, COCO-SSD model is also imported.

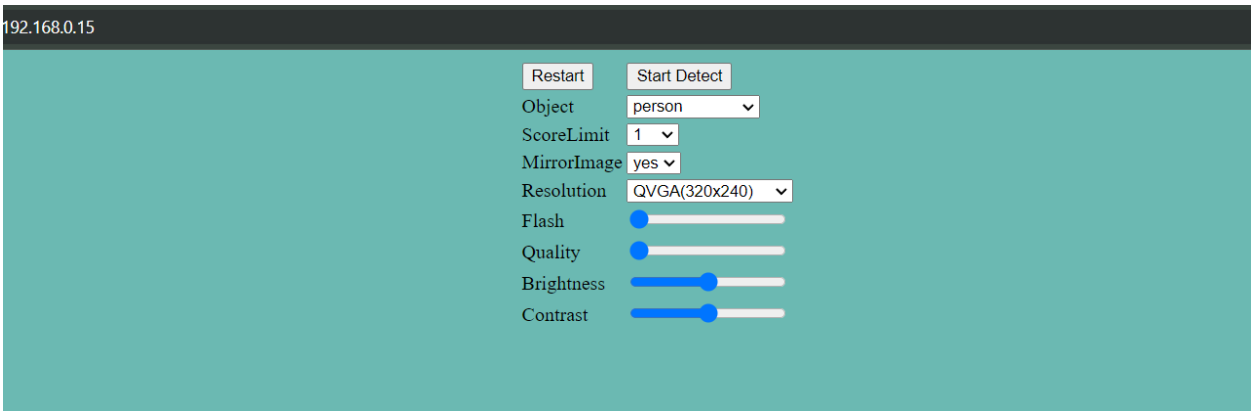


Fig 2: Webpage interface

The webpage includes buttons like Restart to restart the esp32cam board. Start detect button is used to start the Human Detection. The webpage has drop down buttons for selecting the object (Person in our case), ScoreLimit for setting the accuracy score of the detection, MirrorImage to mirror the camera feed, button for setting various resolutions which the ESP342-CAM supports and some drag tabs for setting the flash, quality, brightness and contrast.

WORKING

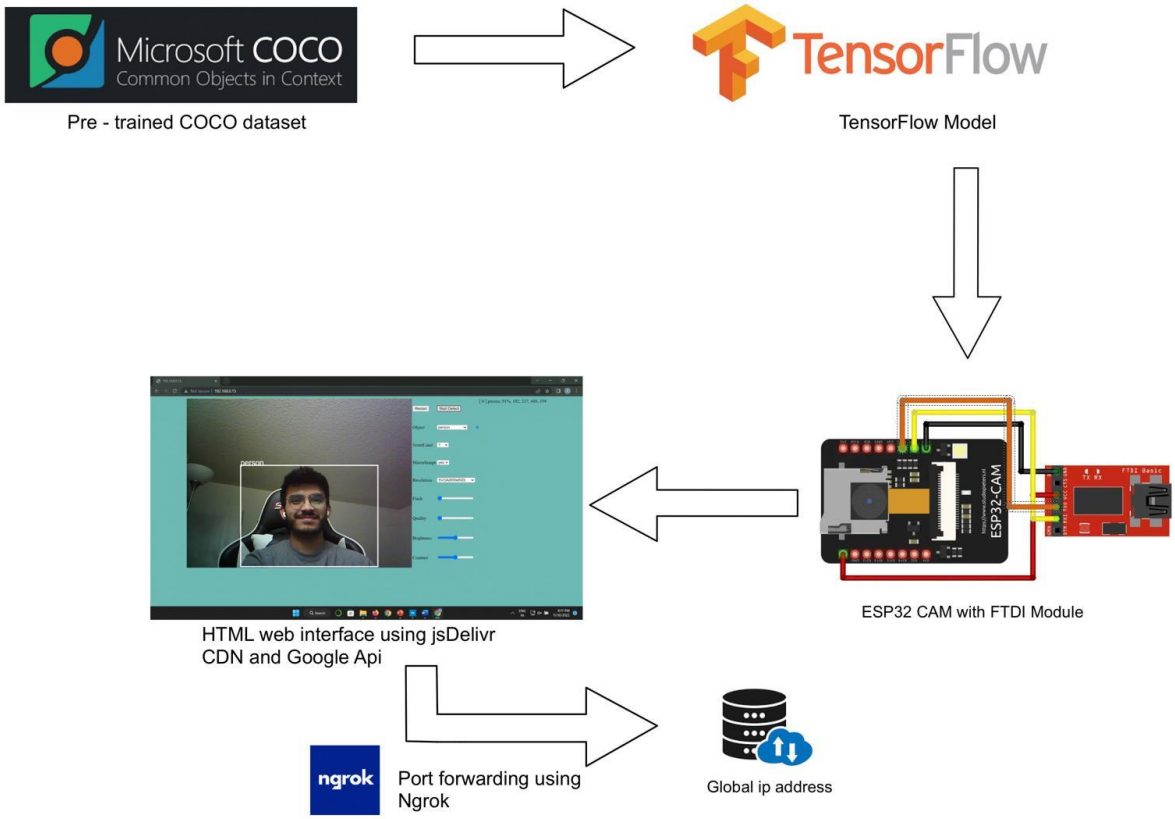


Fig 3: Working architecture of the system

The above figure 3, shows the working architecture of the whole system. Arduino IDE was used to program this project. After calling the necessary API's of TensorFlow model with the COCO-SSD, a person detection function is used which uses the COCO pre-trained weights to identify person from the video feed which is fed to it via the ESP32-CAM. Now, after detecting the person, the system, creates bounding boxes using the labels which it has for better visualization and display output. Usually, the video feed is first split into frames. An array is created of these frames and it is converted into bytes. These are fed to the html and then again decoded for output. Hence, the output is a bit slower than it should be.

## RESULTS AND CONCLUSION

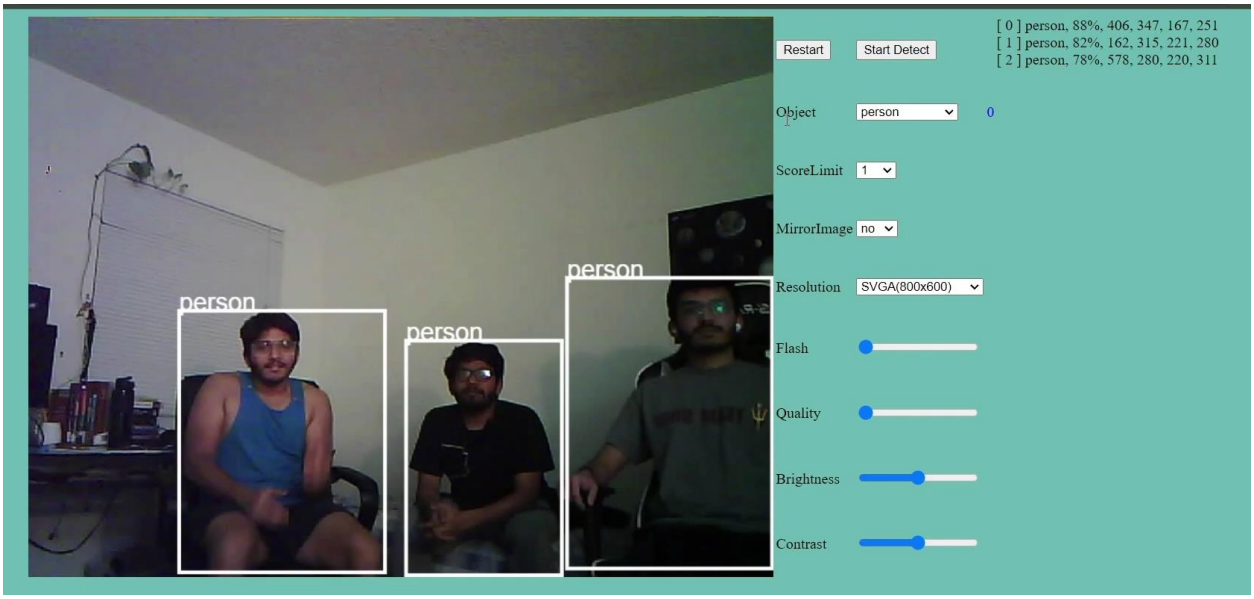


Fig 4: Person detection webpage interface

Figure 4 shows the exact person detection on the webpage which uses custom ip address which is declared earlier. The above detection was carried out in SVGA (800 x 600) pixel resolution with accuracy score 100% so that it detects person in the frames.

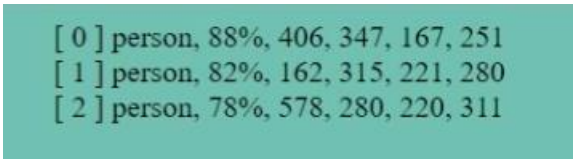


Fig 5: Person counting

The algorithm detects perople and counts them. Figure 5 shows the number of people counted by the system. Such heavy processing takes time and hence sometimes the system gives an error and freezes. Although such a computation time might be reduced with better technology. The trials carried out confirmed the practicality of the suggested framework while showing a plethora of possibilities.

REFERENCES

1. https://docs.platformio.org/en/latest/boards/espressif32/esp32cam.html
2. https://randomnerdtutorials.com/program-upload-code-esp32-cam/
3. Júnior, Manoel José, et al. "Assistive Technology through Internet of Things and Edge Computing." *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, 2019.