REAL-TIME OBJECT DETECTION WITH TENSORFLOW MODEL USING EDGE COMPUTING ARCHITECTURE

DR. MAHIBAN LINDSAY N
Associate Professor
Hindustan Institute of Technology and Science
Chennai, India
nmlindsay@hindustanuniv.ac.in

ALLA ESWARA RAO
Electrical & Electronics Engineering
Hindustan Institute of Technology and Science
Chennai, India
eswararao@ieee.org

MADAKA PAVAN KALYAN
Electrical & Electronics Engineering
Hindustan Institute of Technology and Science
Chennai, India
kalyanmadaka@gmail.com

2022 8th International Conference on Smart Structures and Systems (ICSSS) | 978-1-6654-9761-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSSS54381.2022.9782169

Abstract— This paper presents the capturing of objects using Wi-Fi enabled modular esp32 camera and processes the captured stream of data using machine learning and computer vision techniques, then sends the processed data to the cloud, there are major cloud providers in the market who occupied more than 80% of the global public market the cloud providers are Google Cloud, Amazon AWS, Microsoft Azure. Google Cloud Platform (GCP) is been our primary choice because of its good documentation availability, The Cloud IoT-Core Gateway, as well as a serverless cloud layer to store all of the data. The cloud functions help to trigger the notifications to the users when the cameras detect what we have trained the model. The Edge computing project uses an ESP32 With cameras as a device listener and a raspberry pi as an edge server which has an image classifier model trained with TensorFlow.

Keywords—Edge Computing, IoT, Machine Learning, cloud computing, Google Cloud Platform

I. INTRODUCTION

Edge computing is a framework of distributed computing that brings industrial-grade applications closer to the data generating source like local edge servers. [1] This close contact with data at its source can have significant business advantages. We shifted from using on-premises servers to remote servers, which offer scalability, reduction of the maintenance cost, and an easy usage environment. But for some cases, we need to still cope with processing the data close to where the data is generated. It will provide faster insights, better response, and great bandwidth availability

All around us from our smartwatch to the safety monitoring in the industries and manufacturing sectors edge computing is already in use, in manufacturing sectors to monitor manufacturing processes and apply machine learning techniques and real-time analytics to detect the production errors. A Tesla car can process sensor data that allows it to recognize and self-pilot around pedestrians, and other cars on the road with this computer capability at the edge. [3] [10] pre-loaded mapping data and GPS connectivity are also used to process the sensor data. The edge computing applications are endless. And also, there are several reasons to use edge computing when response time and system availability are critical, even when the internet is unavailable when to use less network bandwidth, privacy, and the considerable part when the processing in the cloud becomes too expensive.

The project aims to fetch a live stream of video from cameras and an on-premises server that searches for cameras using mDNS and uses machine learning [2] [11] and computer vision techniques to analyze the collected stream before uploading it to the Google cloud platform, which will give the ability to stream the footage anywhere anytime and the trigger functions helps triggers alert to the local police station when a person appears with a gun in-store or receive an alert when an unauthorized person tries to enter a home [4] [12], it gives the ability to train custom model and incorporate to the edge computing architecture, the possibilities are endless and up to the end-user creativity and use case of their requirement.

II. PROPOSED METHOD

At an edge computing layer, the processing happens on edge servers which are directly interfaced with a couple to a few thousand and even millions of sensors and controllers, these servers are capable of performing analytics and running machine learning models to make decisions in real-time. [5]
In this project, as a Wi-Fi camera, we're using an ESP32 module with an integrated camera. The processed stream data is securely transferred to Google Cloud Platform utilizing Cloud IoT core using a Raspberry Pi 4B board that acts as an edge server running the TensorFlow object detection model. [6] The data is processed in an event-based way and triggers alerts and a local server provides access to the local web interface to monitor cameras offline, while firebase cloud functions archive the data on firebase to steam the video to internet-connected users on the web interface. [7]

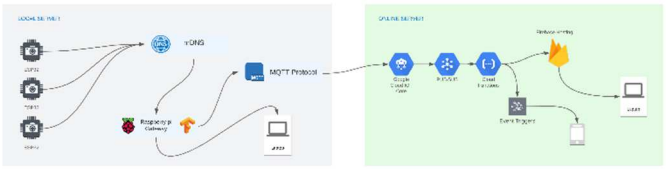

Figure 1 Architecture Overview

The Architecture has two sides Local server-side and Online server-side, local server-side we have cameras as sensors and a local server gateway (raspberry pi), and on the other side, we designed the data flow on a google cloud platform.

A. Flashing ESP32 using PlatformIO

The Esp32 module has a microcontroller of esp32 with an operating frequency of 240MHz and flash memory of 4MB with a ram of 320KB
There are a lot of esp32 modules with an integrated camera in

978-1-6654-9761-9/22/$31.00 ©2022 IEEE

the budget range, In this project, we used the ESP-IDF native programming framework for espressif modules to program the ESP32 firmware. [14]

ESP-IDF is an official IoT development Framework for the ESP32, ESP32-S, and ESP32-C series of SoCs. Which includes a self-contained SDK for developing generic applications on these platforms using programming languages like C and C++. [15] It is not much beginner-friendly to go with so choose to use PlatfromIO which makes it much easier to start with ESP-IDF.

PlatformIO is an integrated development environment (IDE) that offers a collection of cross-platform tools for creating embedded devices. It works with a variety of IoT systems and frameworks. as well as a large number of community-created libraries that may be quickly integrated into your project. It also has a Visual studio code plugin which is very clean and very easy to use.

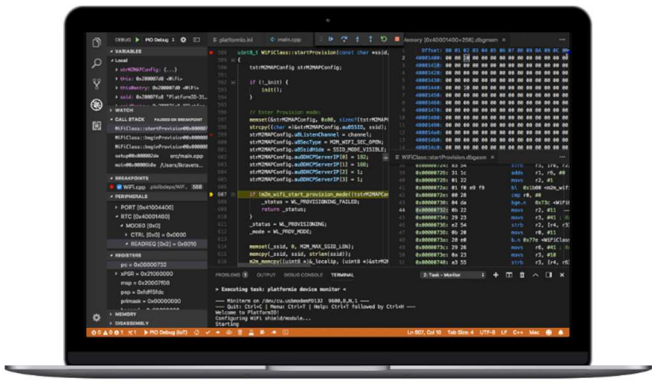

Figure 2 PlatformIO

To flash esp32 we will need an FTDI module to flash the firmware, which can be done by connecting UOT -> RX and UOR -> TX, IO0, IO2 -> GND, and putting the board in the flash mode to be able to program the board. [13]

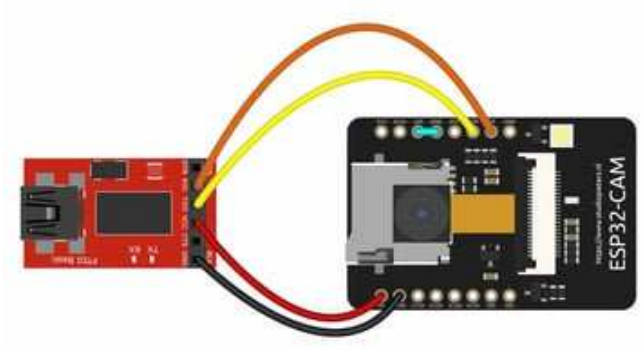

Figure 3 ESP32 And FTDI Module

B. Raspberry pi as Edge Server

The Gateway (Raspberry Pi) searches the network for local cameras using mDNS, detects the objects, and sends the processed data to the cloud, as well as providing a web interface to access the data locally. [8] @tensorflow/models is an NPM package that contains a range of pre-build machine learning models that can be used for a variety of purposes and types of data.

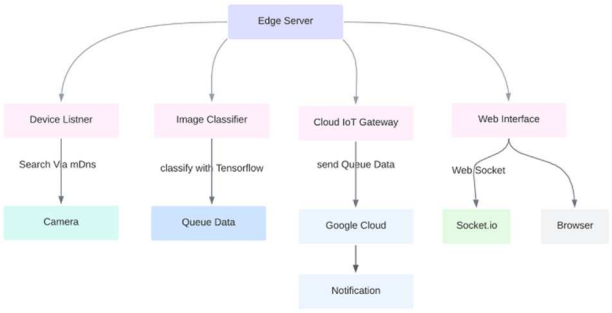

Figure 4 Methodology

Device Listener: Which is responsible to detect cameras in the local network using mDNS and have a record of whether they are online or not.
Image Classifier: TensorFlow is used to receive an image and detect things. we took the CocoSSD pre-trained model found in the tfjs-models package for this project.
Cloud IoT Core Gateway: Responsible to handle all communications and auth with google IoT core
Web Interface: Provide the web interface to view the live stream of the video in the UI on the browser
Edge Server: The edge server is the centralized server here that takes all the above classes to take the input from the camera and pass them through the classifier, sending data to Cloud IoT Core, and displaying the results on the web interface.
Notification: based on the classification result it triggers a message to the user

Build Custom TensorFlow model

Object detection is a computer vision technique that finds the specific class of objects in images and videos. A machine learning approach and a deep learning approach can both be used to detect objects. The machine learning strategy necessitates the definition of features using multiple approaches, followed by classification using any technique, such as Support Vector Machines (SVMs). [9]

Gun violence has been a serious issue all throughout the world, particularly in countries where carrying a gun is permitted, such as the United States. Every year, a large number of innocent people are killed.

Gun violence has killed or injured people in public places, stores, and markets. In this project, we trained our own custom TensorFlow object detection model which detects when a person carries guns in the store the detection helps to trigger the notification alert to the nearby authorities.

There are 4 parts involved in building a custom model, 1) prepare the data 2) Train the model 3) validate the model 4) Deploy the model.

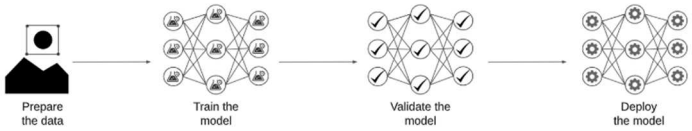

Figure 5 Model Building Flow

1. Prepare the data

The first step to train a great model is to have good quality data. After a bit of online research, we found a few papers that contributed to the world with their collection of Gun Detection datasets, in this project we used Object detection API and train the model using google Collab.

2. Training the model

After having a great set of data now is the time to train the model, training is the state of machine learning where the model is gradually learned from the dataset. In this stage, we need to make the model able to understand the structure of the training dataset which we feed into the model. If we feed with an over amount of train dataset to the model the object detection is not generalized. This problem is called overfitting. [16] It is like we are making the model memorize everything instead of making the model understand how to solve the problem.

To train our model, we used TensorFlow which is an open-source platform for machine learning. The object detection API of TensorFlow makes it easy to construct, train and deploy object detection models.

3. Validate the model

The validation of the model was done by 20% of the total dataset. in the validation, the dataset to evaluate how the model performs against new data it has never seen before.
Precision, Recall, and Loss are three COCO detection evaluation parameters.

We have the values of True_positive (TP) and False_negative (FN) The recall is defined as True_positive divided by the sum of True_positive and False_negative

$$R = \frac{TP}{TP + FN}$$

Precision defines how much you can rely on the positive class prediction it is defined as True_positive divided by the sum of True_positive and False_positive

$$P = \frac{TP}{TP + FP}$$

The precision and recall were divided by Intersection over Union (Io) thresholds. The area of the intersection is divided by the area of the union of a predicted bounding box (Bp) and a ground-truth box (Bt) to get the IoU.

$$IoU = \frac{area(B_p \cap B_t)}{area(B_p \cup B_t)}$$

It's possible to consider that the EU thresholds are used to determine whether detection is a true positive (TP), a false positive (FP), or a false negative (FN).

IoT cloud: Microsoft Azure vs. AWS vs. Google Cloud

Service Providers	AWS VS GCP VS AZURE COMPARISON		
	Key offerings and Functions	Communication Protocols	Edge computing solutions
AWS	AWS IoT core : › Connectivity › Authentication › Rules Engine › Development Environment	› HTTP › MQTT › WebSockets	FreeRTOS Edge Operaating system IoT Green Grass edge computing platform
GCP	Google Cloud IoT core: › Connectivity › Device Management	› HTTP › MQTT	Edge TPU Chip enabling deployment AI at edge
AZURE	Azure IoT Hub: › Connectivity › Authentication › Device Monitoring › Device Management › IoT Edge	› HTTP › MQTT › AMQP Over WebSockets	IoT Edge as an integral part of IoT Hub

The above tables show the comparison of major cloud IoT providers in the current market. The Amazon AWS, Google GCP, and Microsoft Azure. The comparison helps to understand the what service provides with compare to their competitors, we can see that AWS and Azure provide more services than GCP, based on our requirement we can choose the provider.
Microsoft, Google, and Amazon collectively captured 80% of the global IoT public cloud market. Each one has their unique set of features to others

Microsoft Azure IoT
Microsoft's industrial customer base is huge because the company provides the operating system windows is widely used all over the world, it makes clients simple to layer the new Microsoft Azure cloud services on top of current technologies.

Amazon AWS IoT
Amazon's Aws is the world's largest public cloud provider the unique and is the first to offer IoT-specific public cloud solutions in the market making it secure in the first place. As of 2022, the company offers 227 different cloud services

Based on a public IoT case study AWS IoT core is the most popular AWS IoT service, IoT Core for AWS is similar to IoT Hub for Microsoft in terms of functionality. data ingestion service that uses protocols like MQTT or HTTPS to allow secure, bi-directional connectivity for IoT devices. While Microsoft's IoT Hub includes device management capabilities as standard, AWS customers can choose to use IoT Core instead.

GCP IoT
Although Google Cloud is a distant third in the worldwide public cloud computing market, it is the creator of some of the most cutting-edge and widely used cloud technologies. for example, Google created the open-source container orchestration technology Kubernetes. It quickly became the de facto standard for cloud container management.

Pricing Comparison
The feature set is not only the considering parameter when choosing a platform, pricing is also a factor to be considered. The pricing comparison of the major cloud provider gives an overview of how the prices are in the market.
The pricing comparison is done between the AWS, GCP, and AZURE of their IoT services.
The three providers offer the free tier to test out their services more about the pricing comparison is in the bellow table.

Service Providers	AWS VS GCP VS AZURE IoT Price COMPARISON		
	Pricing plan	Free Tire	Free tire across additional IoT services
Amazon IoT core	›Connectivity: $0.08 per million minutes of connection ›Messaging: $0.7-1 per million messages (the more messages the cheaper) ›Device shadow and registry: $1.25 per million operations ›Rules engine: $0.15 per million rules triggered/actions executed	Available for 12 months ›2,250,000 minutes of connection ›500,000 messages ›225,000 registry or Device shadow operations ›225,000 rules triggered and 225,000 actions executed	Available for 12 months ›Device Manageament: 50 remote actions per month ›AWS Greengrass: 3 devices ›AWS IoT Events: 250,000 message evaluations per month ›AWS IoT Analytics: 100 MB of data processes and 10 GB of data storage
Google IoT Control Centre	›$0.00045-0,0045 per MB of data exchanged (the more data the cheaper)	First 250MB Free	›12-month free trial with $300 credit to spend on any Google Cloud Services ›The large suite of always free resources
Azure IoT Hub	›Basic tier: $10 to 500 per unit/per month ›Standard tier: $25 to 2500 per unit/per month ›The price within the tier depends on the number of messages exchanged per day (up to 400,000, 6 million or 300 million)	Up to 8,000 messages per day and up to 500 registered devices	›12-month free trial of popular Azure services ›$200 credit to explore Azure for 30 days ›25+ always free services

4. Deploy the model

Create a protobuf (pb) file from the training checkpoints. This file will contain the graph definition as well as the model's weights. Install the model on the edge server.

C. Google IoT Core Gateway Setup

The Raspberry Pi server functions as a gateway, sending the processed data to the cloud. A gateway is a device that links client devices to Cloud IoT Core and does a variety of tasks. Here it shares the processed data to the cloud.
Configuring the gateway is well documented in the google official tutorial which makes us easy to set up the gateway.

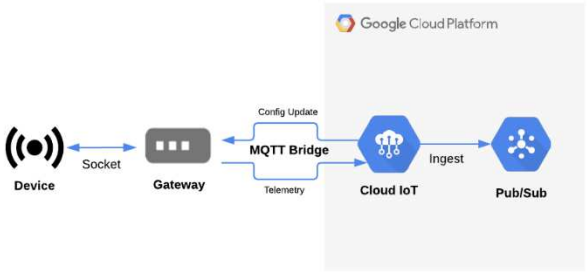

Figure 6: Google Cloud Architecture

To set up the gateway we need to create telemetry and state topics on pubsub in cloud IoT core, for debugging purposes a subscription is required for the Topic. A cloud IoT core registry contains all the registered devise and helps to easily manage them. Cloud IoT core registry Cloud IoT Core and PubSub services can easily be enabled in GCP by searching them in the cloud platform. For registering the device to the cloud, we need key pair certificates by using the certificates we can register the cloud IoT gateway

D. Firebase configuration

The data received by Google Cloud IoT core is processed using Firebase Cloud Functions. These are serverless solutions for cloud code execution. When the Cloud IoT Core submits data to a PubSub topic, a new message is recorded on Firebase Database for that subject.
The Firebase function will listen to both the telemetry and state topics configured on Cloud IoT Core, and then update the data on Firebase

III. CONCLUSION

There are a lot of scenarios out there to deploy this Edge Computing Architecture, in this paper we deployed in a surveillance system that helps to detect any arm caring by a person and sends an immediate alert to the local authorities. We can add as many cameras as we need to this architecture and monitor them seamlessly. The objection detection API works impressively to detect the objects in real-time.

IV. REFERENCES

[1] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," in IEEE Access, vol. 8, pp. 85714-85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[2] Y. Jingyi, S. Rui and W. Tianqi, "Classification of images by using TensorFlow," *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2021, pp. 622-626, doi: 10.1109/ICSP51882.2021.9408796.

[3] C. -H. Hsieh, D. -C. Lin, C. -J. Wang, Z. -T. Chen and J. -J. Liaw, "Real-Time Car Detection and Driving Safety Alarm System With Google Tensorflow Object Detection API," 2019 International Conference on Machine Learning and Cybernetics (ICMLC), 2019, pp. 1-4, doi: 10.1109/ICMLC48188.2019.8949265.

[4] M. Noman, V. Stankovic, and A. Tawfik, "Object Detection Techniques: Overview and Performance Comparison," *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2019, pp. 1-5, doi: 10.1109/ISSPIT47144.2019.9001879.

[5] Mingxing Tan, Ruoming Pang, Quoc V. Le " EfficientDet: Scalable and Efficient Object Detection" Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020), https://doi.org/10.48550/arXiv.1911.09070

[6] Ren, J., Guo, Y., Zhang, D., Liu, Q. and Zhang, Y., 2018. Distributed and efficient object detection in edge computing: Challenges and solutions. IEEE Network, 32(6), pp.137-143.

[7] Wang, Y., Liu, M., Zheng, P., Yang, H. and Zou, J., 2020. A smart surface inspection system using faster RCNN in cloud-edge computing environment. Advanced Engineering Informatics, 43, p.101037.

[8] Hochstetler, J., Padidela, R., Chen, Q., Yang, Q. and Fu, S., 2018, October. Embedded deep learning for vehicular edge computing. In 2018 IEEE/ACM Symposium on Edge Computing (SEC) (pp. 341-343). IEEE.

[9] Pudasaini, D. and Abhari, A., 2020, May. Scalable object detection, tracking and pattern recognition model using edge computing. In 2020 Spring Simulation Conference (SpringSim) (pp. 1-11). IEEE.

[10] Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y. and Shi, W., 2019. Edge computing for autonomous driving: Opportunities and challenges. Proceedings of the IEEE, 107(8), pp.1697-1716.

[11] Li, H., Ota, K. and Dong, M., 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE network, 32(1), pp.96-101.

[12] Rai, P. and Rehman, M., 2019, January. ESP32 based smart surveillance system. In 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) (pp. 1-3). IEEE.

[13] Kaur, A., Jadli, A., Sadhu, A., Goyal, S. and Mehra, A., 2021, November. Cloud Based Surveillance using ESP32 CAM. In 2021 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE) (pp. 1-5). IEEE.

[14] Zare, A. and Iqbal, M.T., 2020, September. Low-cost esp32, raspberry pi, node-red, and mqtt protocol based scada system. In 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS) (pp. 1-5). IEEE.

[15] Shukla, A. and Diwan, R., 2021. IOT Based load Automation with Remote Access Surveillance Using ESP 32 CAM and ESP 8266 Module. Annals of the Romanian Society for Cell Biology, pp.6904-6914.

[16] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks," *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2017, pp. 173-177, doi: 10.1109/INTELCIS.2017.8260032.