Constructing Networked, Intelligent and Adaptable Buildings using Edge Computing

Neha Rudraraju[1] & Kotoju Rajitha[2] & K. Shirisha[3]
[1]Student, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, Telangana, India
[2]Assistant Professor, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, Telangana, India
[3]Assistant professor, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad, Telangana, India

Received: September 14, 2018 Accepted: October 29, 2018

ABSTRACT: *Internet of Things (IoT) devices have created opportunities for integration of physical devices into the digital world, improving efficiency and reducing human exertion. Conventional IoT devices achieve this at the expense of greater latency and hogging down the network bandwidth. In this project we investigate Edge Computing as an alternate to the existing implementations of IoT. After surveying existing models of computing and unique architectures, we employ an IoT system enabling"smart offices" using Fog computing architecture. Our application uses a range of Raspberry Pi devices to achieve this. Other applications such as an Active Badge Location System, employing a digital badge, rather than the conventional physical badge, are alsodiscussed. We conclude by reaping the benefits of edge computing platforms and its impact on the development of prospective IoT applications and the possible future directions.*

Key Words: *Internet of Things, Cloud computing, Edge Computing*

I. Introduction

Cloud computing, over time, has been significantly improved and implemented because of the high-tech and centralized functioning of computing and system management purposes. The accelerated growth and proliferation of IoT and its software is creating a large number of information so compelling it into the cloud isn't feasible. This presents a large challenge to the present cloud computing infrastructure. The significant problems arising are that lots of IoT applications need rigorous latencies and bandwidth. Therefore, we must perform analysis and processing of information close to the end users. IoT also requires assistance for heterogeneity. This is sometimes catered with sub networks. Occasionally resource limitations and network bandwidth limitations arise that need hierarchical processing and offloading attributes. Security of information is also a significant concern. Thus, to realize the entire potential of IoT regardless these issues, a new idea of how Fog/Edge computing has developed.

Fog is an arrangement for distributing computing storage, control, and media services everywhere along the cloud-to-device continuum. Its most important advantages could be illustrated in relation to CEAL in which Cognition(C) suggests that fog is mindful of customer's requirements and will determine where to execute the computing and management functions across the continuum. Fog is effective (E) to make the most of their storage and computation capabilitiesof the resources available on network edge and end-user devices. Agility(A) leaves space for speedy innovation as it is cheaper and quicker to experiment with customer and edge devices. One other important facet is Latency(L) because delay is rapidly reduced by utilizing fog that proves to be beneficial for real timeprocessing andcyber physical systems management.

In the last several decades, miniaturization of technologies has made us credit card sized machines which are more effective than before. There are a whole lot of IoT devices available today. Each one these IoT devices are called to create 5 quintillion bytes of information each and every single day, in an estimated 30 million devices. To upload and process such enormous quantities of information to the dedicated servers is quite a tedious endeavour. Several alternative approaches are suggested to mitigate this dilemma. At Edge Computing we don't send complete information packets to the host, rather the majority of the processing occurs at the border i.e. near the origin where it's being generated and the results are delivered to the host. This raises the efficiency and functioning of the system whilst increasing the throughput at the host. Edge Computing is being widely incorporated due to their increased efficiency of the modern- day CPU's and improved capacity to handle resource intensive tasks with a minimum power usage.

Our Smart Building execution includes three modules:

- Identification of individuals using face recognition

- Identifying and granting access to vehicles, using number plate recognition
- Locating individuals using a Digital Badge Location System

In a true edge computing fashion, all the computation happens at the IoT devices and the centralized server merely acts as an aggregator. This is in contrast to the existing smart building systems, where the IoT devices are used only for sensingand most of the computation happens at a centralized server. We provide a web application, intended for the administrator of the building, to view all the processed information gathered from different devices. These types of smart buildings and offices gather a lot of information about the employees and raise a lot of privacy concerns, which are addressed in the last section.

II.EDGE, FOG, CLOUD: A COMPARISON

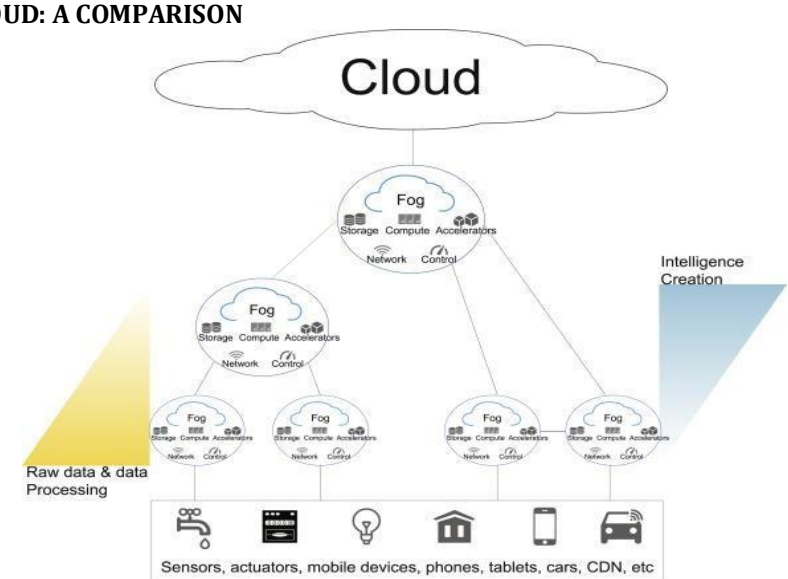

Fig. 1: A typical hierarchical architecture based on fog computing

Edge Computing suggests the concept of making the computing resources available at the edge of the network, that is near the end-devices. The program services are hosted at the network borders such as switches, routers etc. and hence the highest computation occurs within the proximity of the end users. Cisco, on identical lines, described the idea of computing. According to the paradigm, the fog functions as a layer between the border and the cloud that extends the cloud nearer to the nodes which generate and function on IOT data. At times the expression"fog" can be used interchangeably with the word"edge" although fog is a much wider notion.

On the other hand, the idea of fog/edge computing differs from its conventional cloud-based counterpart. It's different from the cloud in the supported computation design, dimensions, infrastructure and software. Cloud is a concentrated platform compared to fog/edge that is a distributed system. They vary in dimensions as cloud information facilities are enormous whereas fog could be of the dimensions needed by the client. The software supported by cloud are for the most part cyber domain whilst fog may also appeal to cyber physiological applications that are time critical. Communication and networking happen close to the end users in fog as opposed to routing traffic through the backbone networks all the way to the cloud.

Fog/Edge can be a complement to the cloud in different ways. This is further justified as fog/edge empowers a service continuum; it fills in the space between the things and the cloud. They are interdependent on each other since fog can work as a proxyof the cloud and supply solutions to the end devices and can function as a proxy of the end points and supply informationto the cloud. Some applications are best suited to be completed at the fog whileothers are more suited to be completed at the cloud.

II. Modules Description

Raspberry-pi Face Recognition

In this module, we have implemented a facial recognition system using raspberry-pi as an edge device. As mentioned previously, edge computing enables connected devices to process data closer to where it is created — or the “edge.” This can be either within the device itself (i.e. sensors), or close to the device, and provides an alternative to sending data to a centralized cloud for processing. Therefore, using raspberry-pi

as an edge device enhances the processing and computing capabilities giving us the speed we desire for recognition.

In the proposed system developed, we assume that we are provided with a camera module that is already integrated with the raspberry-pi and is mounted at the location where we want the faces to be detected and thus gaining access based on the recognitions. We also assume that photographs of the person in six different angles have been taken at the time of commencing work on which processing is done and the output is appended to the embeddings filefor the purpose of future recognitions.

Here, facial recognition can be performed in both images and video streams and it is done using OpenCv, Python and Deep learning. The deep learning based facial embeddings used here are both highly accurate and capable of being executed in real time. In deep learning, we accept a single input image and output a classification/label for that image. But, for our deep learning + face recognition we have made use of deep metric learning. Here, instead of trying to output a single label (or even the coordinates/bounding box of objects in an image) we are outputting a real-valued feature vector. For the dlib facial recognition network, the output feature vector is 128-d (i.e., a list of 128 real-valued numbers) that is used to quantify the face. Training the network is done using triplets. Deep learning has proven itself as a successful set of models for learning useful semantic representations of data. These, however, are mostly implicitly learned as part of a classification task. In deep metric learning, we aim to learn useful representations by distance comparisons. Accordingly, we try to fit a metric embedding S so that:

S(x, x1) > S(x, x2), ∀x, x1, x2 ∈ P for which r(x, x1) > r(x, x2). where r(x, x 0) is a rough similarity measure given by an oracle. We focus on finding an L2 embedding, by learning a function F(x) for which S(x, x 0) = kF(x) − F(x 0)k2.

Our network architecture for face recognition is based on ResNet-34 from "Deep residual learning for image recognition" paper by He et al., but with fewer layers and the number of filters reduced by half.

The network itself was trained by **"Davis King"** on a dataset of ∼3 million images. On the "Labelled Faces in the Wild" dataset the network compares to other state-of-the-art methods, reaching **99.38% accuracy**.

Facial recognition via deep metric learning involves a "triplet training step". The triplet consists of 3 unique face images- 2 of the 3 are the same person. The neural network generates a 128-d vector for each of the 3 face images. For the two face images of the same person, we tweak the neural network weights to make the vector closer via distance metric so that the 128-d measurements of the two same person photographs will be closer to each other and farther from the measurements of the third different person. After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements. Machine learning people call the 128 measurements of each face an **embedding**. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation).

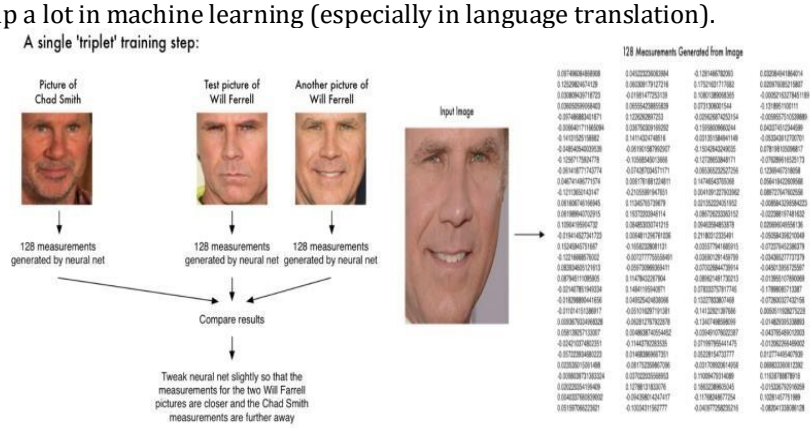

Fig 2: The left figure illustrates the triplet training set and the right figure illustrates the corresponding 128-d measurements for each face on running them through the pre-trained network.

This process of training a conventional neural system to output facial embeddings requires a great deal of data and computational power. But when the network has been trained, it cangenerate measurements for any profile, even for the ones it has never seen before, so this step only needs to be done once. Consequently, in practice we train the system on a PC with a very high computational capability and the embeddings file hence generated is pushed to the raspi, which can be further utilized for face detection. To be able to

perform face recognition with Python and OpenCV we have installed two extra libraries namely dlib along with face recognition. The dlib library, preserved by Davis King, contains our implementation of "deep metric learning" that is utilized to build our face embeddings for the true recognition process. Dlib is a contemporary C++ toolkit comprising machine learning tools and algorithms for creating complex applications in C++ to address real world issues.

The face_recognition command enables you to distinguish faces in a photograph or folder filled with photographs. To begin with, you have to offer a folder with one image of every individual you know. There should be one image file for each individual with the files named according to who is in the picture. Next, you will need another folder with the files containing photographs on which face recognition is to be performed. Then in you just run the commandface_recognition, passing in the folder of known people along with the folder (or only an image) containing unknown people and it tells you who's in each picture.

Steps for performing face recognition:

Step#1: Gather the faces dataset

Before we can apply face recognition we first need to gather our dataset of example images we want to recognize. For our project we choose to collect faces of employees, six each. All of these pictures are stored in the firebase database in JSON format with the name and respective photographs.

Step#2: Compute the embeddings

We have used a deep neural network to compute a 128-d vector (i.e., a list of 128 floating point values) that will quantify each face in the dataset. We begin by capturing input frames from the camera, workflow consists of detecting faces, computing embeddings and in-order to recognise the face we compare the vector to the database via a voting method where we put to use OpenCV, dlib and face_recognition. In our project, computing is done on a powerful GPU machine and the corresponding embeddings file of the faces is stored on the raspi for recognition.

First, we import the required packages and then we handle our command line arguments with argparse:

--dataset : The path to our dataset.

--encodings : Our face encodings are written to the file that this argument points to.

--detection-method : Before we can *encode* faces in images we first need to *detect*them. For raspi we use hog detection method.

Up next, we get the paths to the image files in our dataset and from there we'll loop over each face in the dataset:

1. Extract the person's name from the path.
2. Load and convert the image to rgb.
3. Localize faces in the image.
4. Compute face embeddings and add them to known Encodings along with their name added to a corresponding list element in known Names.

Then, we export the facial encodings to the disk so they can be used in the facial recognition script. On creating facial embeddings via a command, we'll have a pickle file at our disposal- "encodings.pickle" which contains 128-d face embeddings for each face in the dataset.

Step#3: Recognize faces

We grab a frame from the camera, pre-process it. The pre-processing includes resizing followed by conversion to grayscale and rgb. We now compute the 128-d encodings thus quantifying the face. Then we loop over the face encodings folder and check for matches. If matches are found, we'll use a voting system to determine whose face it most likely is. This method works by checking which person in the dataset has the most matches and from there we output the predicted name.

Automatic Number Plate Recognition

Automatic number-plate recognition is a technology which utilizes optical character recognition on pictures to read automobile registration plates. Optical character recognition is the mechanical or digital conversion of pictures of either typed text, or printed text to machineencoded text, from a scanned file, a photograph of a record, a scene-photo (for instance the text on billboards and signs in a photograph) or out of subtitle text superimposed on a picture.

We pick an image and perform optical character recognition, also called optical character read onto it, which utilizes a two-pass strategy to character recognition. Throughout the pre-processing we carry outcharacter isolation or"segmentation" in which to get per-character OCR, multiple characters that are connected due to image artifacts have to be separated; single characters that are broken into multiple pieces due to artifacts must be connected.In the first phase, we carry out feature extraction which decomposes glyphs to"features" including lines, closed loopsand line intersections. The extraction features decrease the dimensionality of

the representation and also makes the recognition procedure computationally efficient. These features are juxtaposed with an abstract vector-like representation of a character, which could decrease to one or more glyph prototypes. Nearest neighbour classifiers like the k-nearest neighbours algorithm have been utilized to compare the image features with the saved glyph features and decide on the closest match. The next pass is called the "adaptive recognition" and utilizes the letter contours recognized with higher confidence on the very first pass to identify better the remaining letters on the next pass. For the interest of post-processing, Tesseract utilizes its dictionary to influence the character segmentation step, for enhanced precision.

For the interest of our job we've incorporated the OpenALPR computer software that's an open source Automatic License Plate Recognition library composed in C++ using bindings from C#, Java, Node.js, along with Python. The library analyses graphics and video streams to discover license plates. The output signal is the text representation of almost any license plate characters.

Digital Badge Location System

The need to locate individuals and staff in an organization arises frequently. Frequently, a mobile phone is used to communicate with the person, but this might not be desirable all the times and sometimes cause disruption to other members. However, a Digital Badge has the advantage of quietly gathering an individual's location.

A `Digital Badge` is an electronic tag that at all times emits a signal which uniquely identifies itself. These signals are then picked up by a network of sensors that are placed throughout the building. These sensors then push the data to a centralized server, where further processing like mapping the unique signal to an individual can happen.

Digital Badges / Active Badges are studied well in the literature. In the two decades, since the conception of an Active Badge, technology has exploded, smartphones became an integral part to modern life, a basic necessity, miniaturization has left us with credit card sized computers (cite Section on Raspberry Pi). We explore Digital Badges in the modern era, and how they fit in well with all the other modules of our Smart Buildings.

What constitutes a Digital Badge is left to the organization, as long as it meets the requirements that constitute a Digital Badge. In this module we detail our implementation of a location system with the following constituents: a Smartphone as a Digital Badge, with the WiFi probe requests as the signal, the MAC address as a unique identifier, and Raspberry Pi's (Nodes) as sensors.

A MAC (Media Access Control) address is a unique identifier that's associated with a Network Interface Controller (NIC) for communications. MAC addresses are used as as a network address for IEEE 802 technologies, including Ethernet, Bluetooth and Wi-Fi. MAC addresses are formed to according to the rules of IEEE, and these are guaranteed to be unique for every networking device, making these our candidate to uniquely identify individuals.

When a device's Wi-Fi is turned on, it starts actively broadcasting small packets of data called Probe requests. Hotels, malls, and airports have been known to collect Probe requests to track the footprint of unknowing passersby. We use these to track the location of individuals. These probe requests contain the MAC address of the sender and we exploit these for our location system.

A survey of existing smartphones show that a device may send anywhere between 55 and 2000 probe requests an hour. Surveys show that both Apple and Android devices, when active sends a probe request every 15 seconds, and every 2 to 5 minutes when inactive (passive scan).

The second part of the system involves a Wi-Fi sniffing device, here we are using a Raspberry Pi 3. The Wi-Fi module onboard the Raspberry device doesn't come with a default monitoring mode, we use a custom firmware provided by the software package nexmon to configure the Wi-Fi unit to run in a monitor mode. In monitor mode, the Wi-Fi module scans the channel for Wireless communications. Setting up the module to scan for probe requests is a trivial process.

Each node maintains a database of registered MAC addresses. When it detects a Probe request with a MAC address that matches up against the one in its database, the Node send its Node Id, the time of capture, and the identified MAC address to the centralized server. The server updates its database with the new information. On the server side, there is a mapping between the each Node Id and metadata about those Nodes. The metadata includes the location of the node. This information is reflected on the admin's web application.

This is leveraging edge computing, i.e. the identification of individuals happen on the distributed nodes (Raspberry Pi's in our case). This has the advantage of not throttling the centralized server, as thousands of Probe requests are sent per minutes in a moderately crowded location.

Digital Badge Application

We built a modular web application for the location system that is integrated into the rest of the smart building application. The application displays a dynamically updating table of individuals identified at various locations together with the last sighted location. In addition to this, the following queries are provided to the admin:

SEARCH (name)

Displays the current location of the user, along with the location history for the past five minutes

LIST (location)

Displays the list of individuals identified at a given location, with a dynamically updating field, and the individual's last sighted times.

COUNT (location)

Approximates the number of individuals currently present near a node.

SUMMARIZE

Displays a summary of list of all the nodes, along with the number of persons present in the vicinity of that node.

SUMMARIZE (name)

Displays the entire history of the person with all their past location history. This is limited to one hour to dispel privacy concerns. All the information is purged after every hour to prevent data harvesting, and respect the privacy of the users.

Data Representation

The data sent from the nodes to the server is in JSON format, with the following structure:

{ "nodeId" : String // The node id of the node sending the data
, "macAddress": String // The Mac Address identified
, "userId": String // The user id associated with the Mac address
, "time": Date // The timestamp of the sighting
}

We use Firebase to store this data, it provides a Real Time NoSQL database, which is ideal for our use case as we generate a lot of data and they have to be reflected in the web app with very low latency.

The current implementation is limited to Wi-Fi enabled devices and only for providing location information. It is easy to adapt this to any other network enabled device, we could have gone for Bluetooth devices, however Wi-Fi is far more pervasive and consumes less power and most users have it switched on more than Bluetooth. An organization may also make their own custom Digital Badges that communicate via radio signals which would consume a lot less power. Infrared communication has been exploited for a long time and it is inexpensive to implement. Our implementation has the advantage of being low-key and unobtrusive, with possibility of further enhancements like measuring footfall in a particular location.

Digital Badges can be combined with other smart building functions such as: air conditioning, fire alarms, and security. Digital Badge extends the concept of a smart building to take into the consideration the location of personnel in the environment.

IV. Testing

1. Test cases and Results for Face recognition

A. Giving an image of only one person to the recognize_faces.py script:

Input:

$ python3 recognize_faces_image.py–encodings encodings.pickle –image examples/151-2.jpg
[INFO] loading encodings…. [INFO] recognizing faces……. **Output:**
['ian_malcolm']

B. Giving animage having multiple people to the recognize_faces.py script:

Input:

$ python3 recognize_faces_image.py –encodings encodings.pickle –image examples/151-4.jpg
[INFO] loading encodings…. [INFO] recognizing faces……. **Output:**
['ian_malcolm','owen','ellie_sattler']

2. Test cases and Results for Automatic number plate recognition

A.Using the Cloud-API

Input:

$ python .\alpr.py C:\Users\rohan\Desktop\IMG_0329.jpeg **Output:**
{

"uuid": "",
"data_type": "alpr_results",
"epoch_time": 1540736441077,
"processing_time": {
"total": 729.9029999994673,
"plates": 241.89402770996094,
"vehicles": 418.79600001266226
},
"img_height": 2048,
"img_width": 1536,
"results": [
{
"plate": "TS11EF6902",
"confidence": 93.74383544921875,
"region_confidence": 0,
"vehicle_region": {
"y": 777,
"x": 595,
"height": 433,
"width": 433
},
"region": "",
"plate_index": 0,
"processing_time_ms": 39.89396667480469,
"candidates": [
{
"matches_template": 0,
"plate": "TS11EF6902",
"confidence": 93.74383544921875
}
],
"coordinates": [
{
"y": 1052,
"x": 753
},
],
"vehicle": {
"color": [
{
"confidence": 88.48767852783203,
"name": "silver-gray"
},
{
"confidence": 10.567081451416016,
"name": "gold-beige"
},
…. …. …..
],
"make": [
{
"confidence": 77.21997833251953,
"name": "maruti-suzuki"
},
{
"confidence": 22.777862548828125,
"name": "suzuki"

},
….. ……. ……
],
"body_type": [
{
"confidence": 91.19192504882812,
"name": "sedan-compact"
},
{
"confidence": 3.8682174682617188,
"name": "sedan-wagon"
},
…. … ….
],
"year": [
{
"confidence": 89.92891693115234,
"name": "2015-2019"
},
{
"confidence": 10.069289207458496,
"name": "2010-2014"
},
…. …. ……
],
"make_model": [
{
"confidence": 39.05268859863281,
"name": "maruti-suzuki_ertiga"
},
{
"confidence": 34.20917892456055,
"name": "maruti-suzuki_baleno"
},
….. …. …..
]
},
"matches_template": 0,
"requested_topn": 10
}
]
}

Number plate found: **TS11EF6902**

B.Simply running through the openalprlibrary Input:

$.\alpr.exe -c in C:\Users\rohan\Desktop\Tensorflow-License-Plate-Detection\png_tesseract\test_tesseract\image3.jpg **Output:**

plate0: 7 results
- TN21AQ1114 confidence: 84.5511
- TNZ1AQ1114 confidence: 79.3356
- TN2IAQ1114 confidence: 79.0931
- TN21AO1114 confidence: 78.4321
- TN21A01114 confidence: 78.3636
- TN21AD1114 confidence: 77.6253
- TN21AB1114 confidence: 75.2006

V.User Manual

1 Face recognition

Fig 3: Dataset of a particular person

Fig 4: Test image taken for performing face recognition

2. Automatic Number Plate Recognition

Fig 5: Creating the encodings.pickle file and using that to recognize faces in the given test image

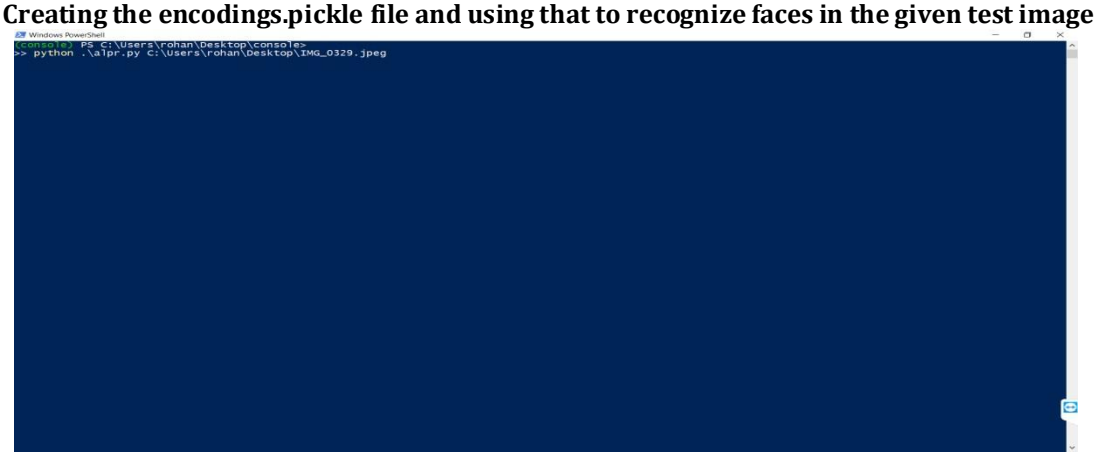

Fig 6: Command to perform anpr using the Cloud-API

e ISSN 2348 –1269, Print ISSN 2349-5138
Cosmos Impact Factor 4.236

Fig 7: Input image for testing

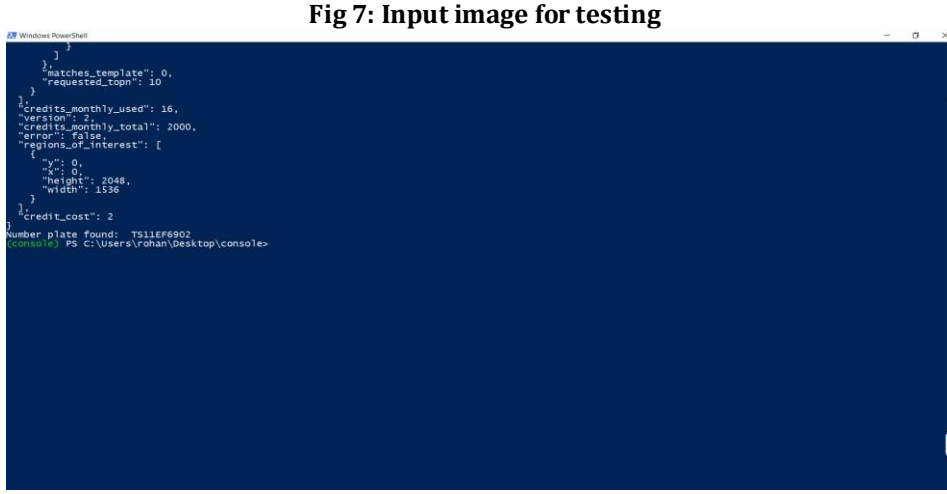

Fig 8: Output for automatic number plate recognition

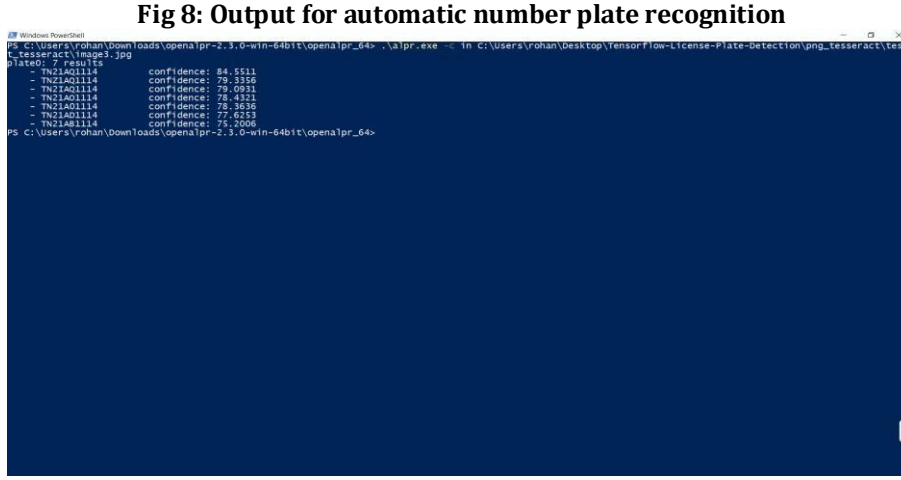

Fig 9: Command and Output on performing anpr using openalpr library in a step-wise manner

Fig 10: Input test image

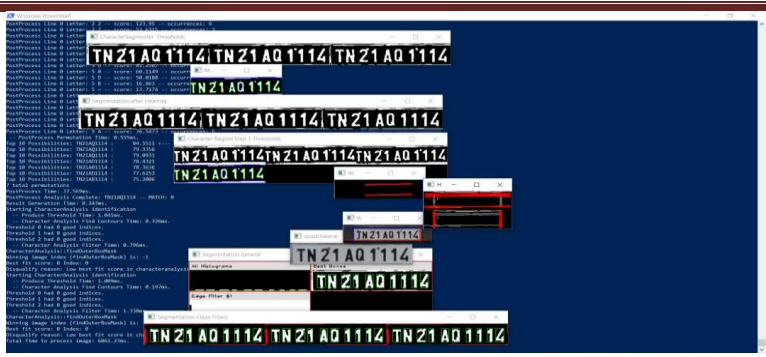

Fig 11: The pipeline stages executing in order

Fig 12: Output on performing character segmentation

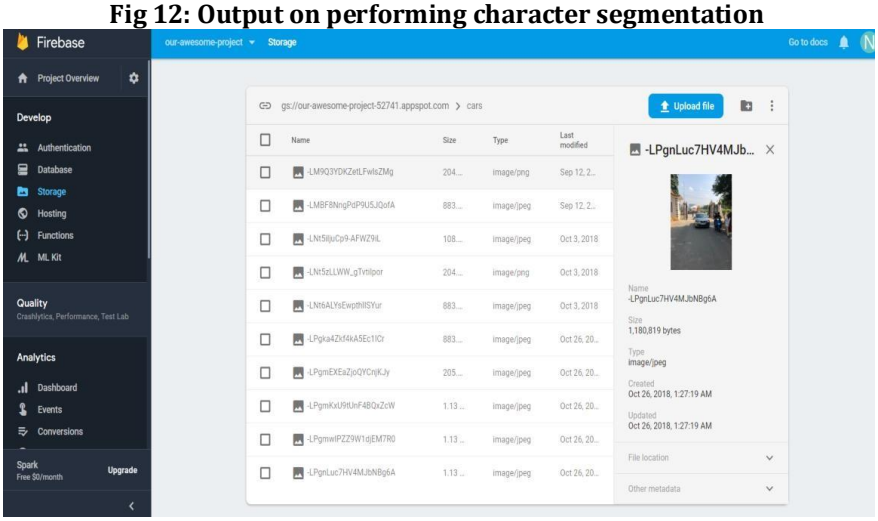

Fig 13: Storing information about the recognized vehicles in the firebase storage

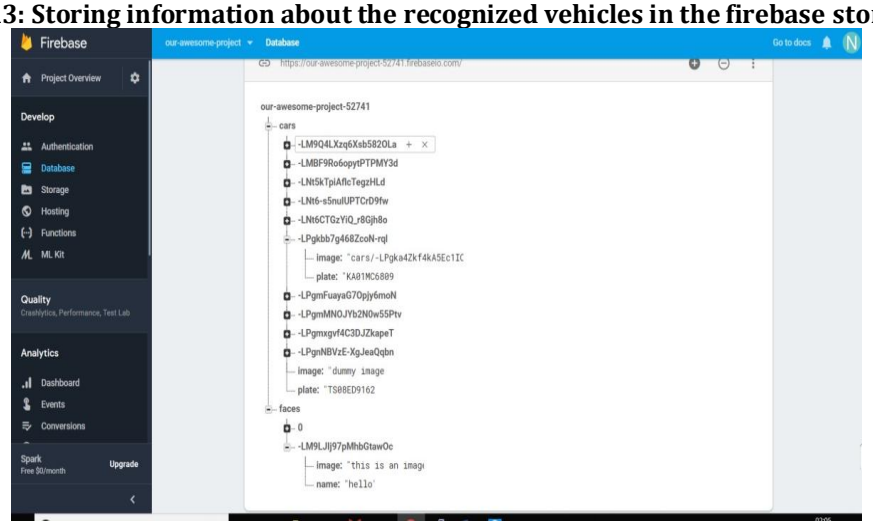

Fig 14: The manner in which recognized face images and vehicle information are pushed onto the firebase database

3. MAC Addresses Detection:

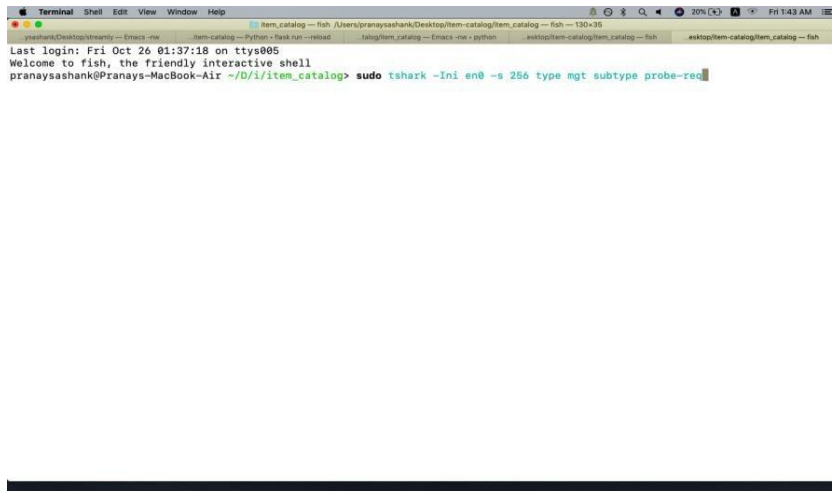

Fig 15: Sniffing the wi-fi for the MAC addresses

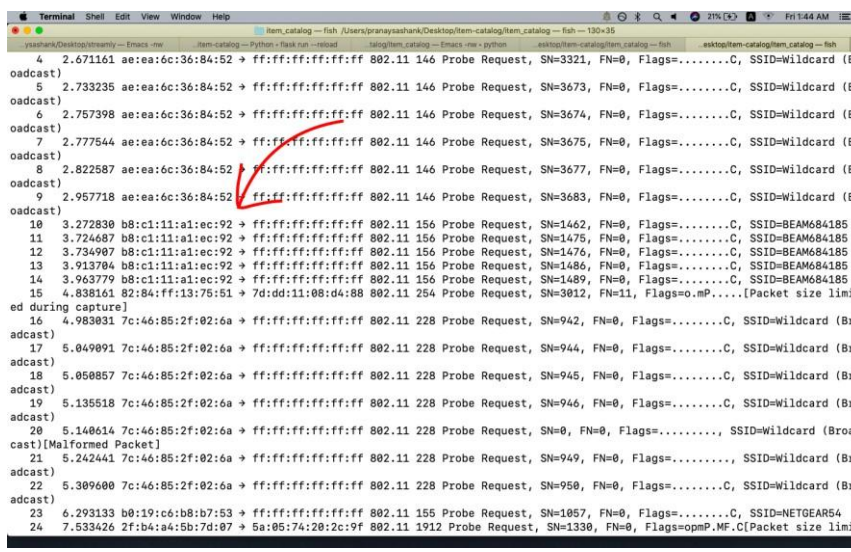

Fig 16: List of identified MAC addresses

3. User Interface

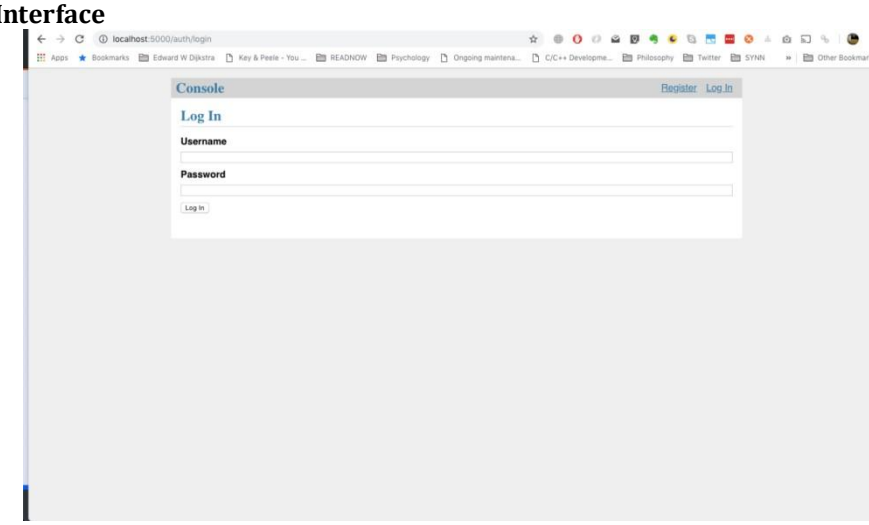

Fig 17: Login page

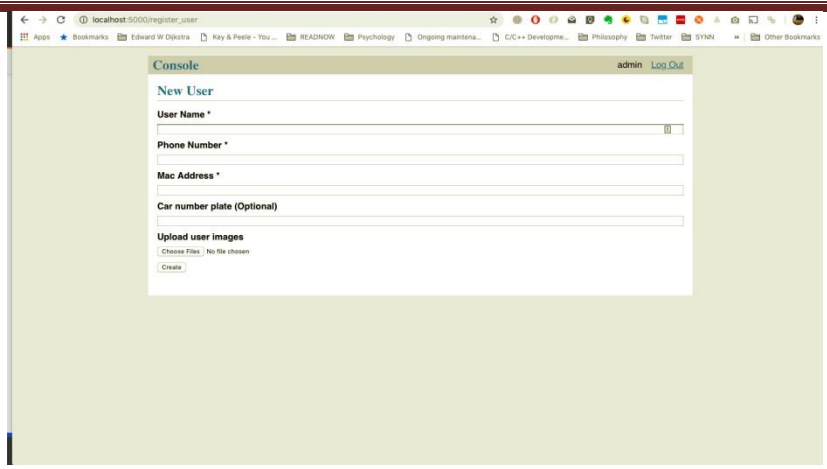

Fig 18: Registering the employees

VI. Conclusion

The Smart Building system we have developed is widely useful and can be used on a daily basis for either logging in people for the day, or locating them in the building, or finding all the vehicles that have entered in a day. We have found the accompanying web application that we have developed to be tremendously useful. It gathers information from different modules and provides it in a user friendly manner. It is useful to take a stock of the current day. Most of the data we gather is already being gathered on a day-to-day basis and as such we are not infringing on users privacy. However all the data gathered is only accessible to the person with the master password and no one else. In addition to this, location information is deleted every hour to prevent abuse of this technology by unscrupulous employers.

One interesting area that we haven't dealt with in our project is Smart Parking. Smart Parking fits well in the overarching idea of IoT. Even though we haven't dealt with how they would be implemented, it's easier to see the benefits. Smart Parking reduces the time spent looking for a parking space, thereby greatly increasing convenience of vehicle users.

Number Plate recognition system could be used in a Smart Building with electronic gates to automate opening and closing the gates based on the number plate recognized. This can then be integrated with Smart Parking

Our Digital Badges is a low-cost solution, we have used readily available smartphones and their MAC addresses for location provenance. This could be replaced with custom built, low power consuming sensors that use Bluetooth or Infrared communication. An interesting further development could be integrating Digital Badges together with facial recognition, for increased accuracy of the location system.

The last few decades have seen enormous growth in processor speeds and research has yielded more efficient algorithms than ever. Combined with the ever increasing efficiency of modern CPUs and cutting edge machine learning algorithms, there has never been a better time to demonstrate the potential of IOT. What we have explored is less than a fraction of what is possible with modern technology. Given, sufficient resources a lot of factors could be fine tuned, from custom chips, to fine tuned algorithms. It is time to envision a new age of "Smart Buildings" that are smarter and better than ever before. The future belongs to IOT and it is indeed very exciting.

References

1. M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," IEEE Internet Things J., vol. 3, no. 6, pp. 854–864, Dec. 2016
2. J. Pan and J. Mcelhannon, "Future edge cloud and edge computing for internet of things applications," IEEE Internet Things J, vol. PP, no. 99, pp. 1 – 1, 2017
3. Roy Want, Andy Hopper, Veronica Falcão and Jonathan Gibbons, "The Active Badge Location System"
4. https://www.github.com
5. https://www.pyimagesearch.com
6. https://www.google.com
7. https://www.wikipedia.org