**MACHINE LEARNING APROACH TO MONITOR DEPARTURE OF BUSES NEAR THE BUSBAY**

**A mini-project report submitted to the JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD in partial fulfillment of the requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

IN

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

K SAI KRISHNA 15071A0520

R MAHENDHAR 15071A0542

Y ABHINAY 15071A0552

K SAI TEJA 15071A0554

R NAVEEN KUMAR 15071A0559

Under the Guidance Of

Dr. S.Nagini



**Computer science and engineering Department**

**VNR Vignana Jyothi Institute of Engineering and Technology**

**(Affiliated to J.N.T.University, Hyderabad)**

**Bachupally(M), Hyderabad, Telangana, India.**

**CERTIFICATE**

This is to certify that K SAI KRISHNA(15071A0520), R.MAHENDHAR(15071A05542) Y ABHINAY(15071A0552),K SAI TEJA(15071A0554), R NAVEEN KUMAR(15071A0559), have successfully completed their mini-project work at CSE Department of VNR VJIET, Hyderabad entitled **“MACHINE LEARNING APROACH TO MONITOR DEPARTURE OF BUSES NEAR THE BUSBAY”** in partial fulfilment of the requirements for the award of B. Tech degree during the academic year 2018-2019. This work is carried out under my supervision and has not been submitted to any other University/Institute for award of any degree/diploma.

Dr.S.Nagini Mrs. B.V.Kiranmayee

CSE Department Head of CSE Department

VNRVJIET VNRVJIET

**ii**

**DECLARATION**

We, here by declare that the project entitled “**MACHINE LEARNING APROACH TO MONITOR DEPARTURE OF BUSES NEAR THE BUSBAY”** is carried out under the guidance of Mrs S.NAGINI is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. This is a record of bonafide carried out by me. The results embodied in the project report have not been submitted to any other university or institute for the award of any other degree or diploma.

K SAI KRISHNA 15071A0520

R MAHENDHAR 15071A0542

Y ABHINAY 15071A0552

K SAI TEJA 15071A0554

R NAVEEN KUMAR 15071A0559

iii

**ACKNOWLEDGEMENT**

First and foremost, I would like to express our deep and sincere gratitude towards our great institution, VNR Vignana Jyothi Institute of Engineering and Technology for providing us with opportunity of fulfilment of our long-cherished goal of becoming BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.

We are very much thankful to our Principal, **Dr. C. D. Naidu** and our Head of Department Mrs. **B. V. Kiranmayee**, for extending their cooperation in doing this project in stipulated time.

We extend our heartfelt thanks to our guide, **Dr. S. Nagini** and the project coordinators **Ms. P. Radhika**, **Ms. R. Vasavi** and **Ms. M. Manasa Devi** for their enthusiastic guidance throughout the course of our project.

Last but not least. Our application obligation also goes for all teaching and non teaching staff members of Computer Science and Engineering Department and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation. We would also like to thank our friends, whose direct or indirect help has enabled us to complete this work successfully.

iv

LIST OF CHAPTERS

CHAPTERS PAGENO

1. ABSTRACT 01
2. EXISTING SYSTEM AND PROPOSED SYSTEM 03
   1. EXISTING SYSTEM 03
      1. SYSTEM ARCHITECTURE 03
      2. DRAWBACKS 04
   2. PROPOSED SYSTEM 04
      1. METHODOLOGY 04
3. FEASIBILTY STUDY 05
   1. ECONOMIC FEASIBILITY 05
   2. TECHNICAL FEASIBILITY 05
   3. OPERATIONAL FEASIBILITY 05
4. SYSTEM ANALYSIS 05
   1. REQUIREMENT ANALYSIS 05
   2. REQUIREMENT SPECIFICATION 06
      1. DATA 06
      2. HARDWARE AND SOFTWARE 06
      3. TRAINING 06
         1. WHY CAFFE MODEL 07
      4. HARDWARE 07
      5. SOFTWARE 07
         1. OPENCV 08
         2. PYTHON INSTALLATION 09

AND OPENCV INSTALLATION

v

* + - 1. DEEP NEURAL NETWORK MODULE 12
         1. BLOB 12

4.2.5.3.1 BLOB EXTRACTION 12

4.2.5.3.2 BLOB REPRESENTATION 13

4.2.5.3.3 BLOB CLASSIFICATION 13

4.2.6 FUNCTIONAL REQUIREMENTS 16

4.2.6.1 CAMERA MODULE 16

4.2.6.2 ALGORITHM 16

5. SOFTWARE DESIGN 16

5.1 MODULES 16

5.2 USE CASE DIAGRAM 17

5.3 CLASS DIAGRAM 18

5.4 SEQUENCE DIAGRAM 19

5.5 ACTIVITY DIAGRAM 20

5.6 WORKFLOW 21

6. IMPLEMENTATION 22

6.1 ENVIRONMENT 22

6.2 MODEL DEVELOPMENT AND TRAINING 22

6.2.1 NEURAL NETWORKS 23

6.2.1.1 CONVOLUTION 23

6.2.1.2 MAX POOLING 24

6.2.1.3 ACTIVATION 25

vi

6.2.1.4 FLATTEN 25

6.2.1.5 DENSE 26

6.2.1.6 DROPOUT 26

6.2.1.7 CONVOLUTIONAL NEURAL NETWORKS WITH 27

ALL CONNECTED LAYERS

6.3 TRAINED MODEL 28

6.4 PREDICTION 28

7. TESTING 29

8. RESULTS 33

8.1 ACCURACY 34

9. CODE 34

10. COCLUSION 40

11. FUTURE SCOPE 40

12. REFERENCES 40

vii

**ABSTRACT**

Buses are the lifeline of the city transport. Especially in a developing nation like India, public transport is often not a matter of choice but an absolute necessity. With the adventof public transport in our country many people started using it as it is a reliable and cheap mode of transportation. This helps in building communities and ensures that people from all social backgrounds and locations can get to work, schools and colleges. It also helps to access vital services and become their sole medium of transportation which provides access to employment, community resources, medical care and recreational opportunities. It’s no exaggeration to say that without buses, many towns and cities would grind to a halt. Using public transport helps reduce congestion, pollution and fuel consumption which in turn will build a better future for our next generation.

Currently TSRTC has three zones, Hyderabad, Greater Hyderabad and Karimnagar. It has 13 regions, 97 depots and 357 bus stations. TSRTC buses undertake operations on 3,687 routes, with an approximate fleet of 10,479 under its wing carrying around 50 lakhs passengers to their destination every day. Nowadays majority of the people prefer other means of transportation instead of traditional RTC transportation. The lack of reliable and safe means of transportation is clearly mentioned as one of the contributing factors leading to frustration and anger in a large number of people. The population and the bus ratio has also become a major problem. An unfriendly public transport system that is not adapted keeping the special needs of the public in mind reduces their options forcing them to settle with a less-productive local employment opportunities. Improper halt and speeding up harshly at the bus stop are some of the primary problems that the passengers face these days. Due to this, Women, children, Senior citizens and physically challenged passengers are not able to board the buses with ease and we also observe that most of the youngsters are risking their lives while

1

running behind the buses to catch it. It has become a boundless problem and it must be figured out for societal benefit and to extend better services to the public.

RATING

One of the solutions to tackle this pain point is through monitoring the halt time of RTC buses at bus stop. This system consists of two cameras, one at the start point and another at the end point of the bus stop which uniquely identifies each RTC bus and gets all the information about the bus from the database. This calculates the entire waiting time of the bus at a particular stop. If the bus doesn’t halt at the bus bay, an appropriate measure is taken thus providing a safe boarding. With this we can increase the robustness and credibility of the system by using machine learning algorithms and GPS.

This methodology if implemented will streamline the means of transport thereby making it as the most reliable source that would certainly augment revenue generation of TSRTC.

**2**

**EXISTING SYSTEM AND PROPOSED SYSTEM**

**2.1 Existing system**

**Bus Recognition System for Visually Impaired Persons Using RF Module**

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

a bus detection system using

RFID technology that aims to ease the traveling and movement

of blind people. The proposed system consists of two detection

subsystems, one on the buses and the other on the bus stations,

database system and a website. In the bus detection subsystem,

the nearby stations will be easily detected and then announced

through a voice message inside the bus. Moreover, any existing

blind person in the surrounding area of the station will be

detected by the bus subsystem to alert the bus driver about the

number of blind persons. In the bus station subsystem, the

coming buses will be detected and then announced in the station

in order to alert the blind people. A complete system prototype

has been constructed and tested to validate the proposed system.

The archived results show that the system performance is

promising in terms of system functionality, safety, and cost.

A bus detection system using RFID technology that aims to ease the traveling and movement of blind people. The proposed system consists of two detection subsystems, one on the buses and the other on the bus stations, database system and a website. In the bus detection subsystem, the nearby stations will be easily detected and then announced through a voice message inside the bus. Moreover, any existing blind person in the surrounding area of the station will be detected by the bus subsystem to alert the bus driver about the number of blind persons. In the bus station subsystem, the coming buses will be detected and then announced in the station in order to alert the blind people. A complete system prototype has been constructed and tested to validate the proposed system. The archived results show that the system performance is promising in terms of system functionality, safety, and cost.

**2.1.1 System Architecture :**

The architecture of system is based on a distributed model as shown in the figure 1.This system mainly consists of two parts: Bus transceiver segment and VIP transceiver segment. Through the device of the VIP person, one can set the bus number which he/she wants to board. Once the device is switched ON the transmitter of the device will start radiating the information up to some distance of radius, say 10 m. The buses in the vicinity will then respond to him/her through another transceiver installed and thus the VIP will be acknowledged.



**Fig: ARCHITECTURE**

**3**

****

**Fig: Illustration of the working of VIP’s Personal Digital Assistant**

**2.1.2Drawbacks:**

This system is limited to only blind people.

**2.2 Proposed system**

This system helps to find out waiting time of TSRTC bus near the bus stop thus helping to safely board the passengers

**2.2.1 Methodology**

This system consists of a Camera near the bus stopwhich identifies each TSRTC bus usingimage classification algorithms to detect the bus andsave it as a matObject which will be processed bydeep neural networks to detect the bus which is usedfor calculating the in-time

4

and out-time of the bus from the bus bay.So that we can estimate wheather the paasengers are boarding the bus safely or not.

**3 Feasibility Study**

**3.1 Economic feasibility**

The cost of creating the model might be high depending on the type of resources it requires. The operational cost will be less once the model becomes operational since the only expenditure incurred while running the system would be the cost of maintaining the camera module safely near the bus stop. Most of the money would be spent on research and development of the machine learning model.

**3.2 Technical feasibility**

All the technical requirements can be easily met since most of the softwares are open source and we can easily use the softwares with a system having a good processing RAM.

**3.3 Operational feasibility**

It is quite simple to operate because after installing the camera module in the bus stop no extra work is required but monitoring of the modules should be done frequently because any minute damages may lead to change in the results. The users of the system don’t need any training on the system.

**4 System analysis**

**4.1 Requirement analysis**

The whole system is entirely automated when it is completely deployed and the system does not require any human work once it is correctly deployed. Only thing is it has to be trained

5

well and the system has to be cross checked at starting. The main part of the project is we have to train the model so that it gives an accurate result at any condition.

**4.2 Requirement specification**

**4.2.1 Data**

Data is an essential part of the project and without it project cannot be done. The data we require is supposed to be in DICOM format and a large sample is required to train the model. We have taken a caffe model and trained the model to have the system to detect the bus and give the desired results.

The data should be given to the system and the system is trained in such a way that the system should recognise the bus even if the frame consists of high noise.

**4.2.2 Hardware and Software**

A decent amount of hardware is required to create and implement the project at different phases and environment.

**4.2.3 Training**

To train the machine on huge amounts of data normal systems are not sufficient a decent GPU with a computational capability score of 6.1 is required and with at least 4gb of RAM. These requirements are the bare minimum and in our case we had NVidia gtx 1050 with 8gb of RAM . Having a much better GPU can significantly speed up the training process.

We used caffe model for training the system. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research ([BAIR](http://bair.berkeley.edu/)) and by community contributors. [Yangqing Jia](http://daggerfs.com/) created the project during his PhD at UC Berkeley. Caffe is released under the [BSD 2-Clause license](https://github.com/BVLC/caffe/blob/master/LICENSE).

**6**

**4.2.3.1 Why caffe?**

**Expressive architecture** encourages application and innovation. Models and optimization are defined by configuration without hard-coding. Switch between CPU and GPU by setting a single flag to train on a GPU machine then deploy to commodity clusters or mobile devices.

**Extensible code** fosters active development. In Caffe’s first year, it has been forked by over are defined by configuration without hard-coding. Switch between CPU and GPU by setting a single flag to train on a GPU machine then to deploy to commodity clusters or mobile devices.

**4.2.4 Hardware**

Our system needs a very decent camera module with a minimum of 2 mega pixel lens to detect the bus and we have to also deploy the cameras with protection so that they are not affected by the weather like rain and heat.

The camera module should be able to record the live video from the bus bay and that video captured by the camera module is used for calculating the halt time of bus near the bus bay

**4.2.5 Software**

In order to meet all the above-mentioned software requirements we used certain packages that speeded up the environment setup process. The following softwares were used in the entire project

* Opencv 3.3.
* Python 3.6.
* Deep neural networks.

**7**

**4.2.5.1 Opencv 3.3:**

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

**Why opencv?**

## **Specific**

OpenCV was made for image processing. Each function and data structure was designed with the Image Processing coder in mind. Matlab, on the other hand, is quite generic. You get almost anything in the world in the form of toolboxes. All the way from financial toolboxes to highly specialized DNA toolboxes.

## **Speedy**

Matlab is just way too slow. Matlab itself is built upon Java. And Java is built upon C. So when you run a Matlab program, your computer is busy trying to interpret all that Matlab code. Then it turns it into Java, and then finally executes the code.

If you use C/C++ you don't waste all that time. You directly provide machine language code to the computer, and it gets executed. So ultimately you get more image processing, and not more interpreting.

I've tried doing some real time image processing with both Matlab and OpenCV. I usually got very low speeds, a maximum of about 4-5 frames being processed per second. With OpenCV, I get actual real time processing at around 30 frames being processed per second.

Sure you pay the price for speed - a more cryptic language to deal with, but its definitely worth it... You can do a lot more... you could do some really complex mathematics on images with C and still get away with good enough speeds for your application.

8

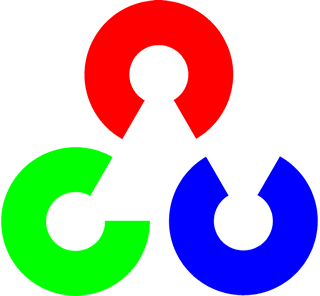


Fig: OpenCv

## **Efficient**

Matlab uses just way too much system resources. With OpenCV, you can get away with as little as 10mb RAM for a real time application. But with today's computers, the RAM factor isn't a big thing to be worried about. You do need to take care about memory leaks, but it isn't that difficult.

**4.2.5.2 Python Installation**

* Python can be obtained from the **Python Software Foundation** website at [python.org](https://www.python.org/). Typically, that involves downloading the appropriate **installer**for your operating system and running it on your machine.
* Some operating systems, notably Linux, provide a **package manager** that can be run to install Python.
* On macOS, the best way to install Python 3 involves installing a package manager called **Homebrew**. You’ll see how to do this in the relevant section in the tutorial.
* On mobile operating systems like Android and iOS, you can install apps that provide a Python programming environment. This can be a great way to practice your coding skills on the go.

9

Alternatively, there are several websites that allow you to access a Python interpreter online without installing anything on your computer at all.

**Note:** There is a chance that Python may have been shipped with your operating system and is already installed. Even if that is the case, it may be that the installed version is outdated, in which case you will want to obtain the latest version anyhow.

In this Python installation guide, you’ll see step by step how to set up a working Python 3 distribution on Windows, macOS, Linux, iOS, and Android. So let’s get started!

## Windows

It is highly unlikely that your Windows system shipped with Python already installed. Windows systems typically do not. Fortunately, installing does not involve much more than downloading the Python installer from the [python.org website](https://www.python.org/) and running it. Let’s take a look at how to install Python 3 on Windows:

### Step 1: Download the Python 3 Installer

1. Open a browser window and navigate to the [Download page for Windows](https://www.python.org/downloads/windows/) at [python.org](https://www.python.org/).
2. Underneath the heading at the top that says **Python Releases for Windows**, click on the link for the **Latest Python 3 Release - Python 3.x.x**. (As of this writing, the latest is Python 3.6.5.)
3. Scroll to the bottom and select either **Windows x86-64 executable installer**for 64-bit or **Windows x86 executable installer** for 32-bit. (See below.)

#### Sidebar: 32-bit or 64-bit Python?

For Windows, you can choose either the 32-bit or 64-bit installer. Here’s what the difference between the two comes down to:

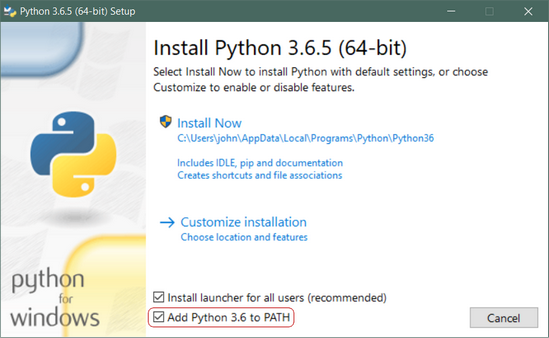
* If your system has a 32-bit processor, then you should choose the 32-bit installer.
* On a 64-bit system, either installer will actually work for most purposes. The 32-bit version will generally use less memory, but the 64-bit version performs better for applications with intensive computation.
* If you’re unsure which version to pick, go with the 64-bit version.

**Note:** Remember that if you get this choice “wrong” and would like to switch to another version of Python, you can just uninstall Python and then re-install it by downloading another installer from [python.org](https://python.org/).

### 10

### Step 2: Run the Installer

Once you have chosen and downloaded an installer, simply run it by double-clicking on the downloaded file. A dialog should appear that looks something like this:

[](https://files.realpython.com/media/win-install-dialog.40e3ded144b0.png)

**Important:** You want to be sure to check the box that says **Add Python 3.x to PATH** as shown to ensure that the interpreter will be placed in your execution path.

Then just click **Install Now**. That should be all there is to it. A few minutes later you should have a working Python 3 installation on your system.

**Installation of opencv:**

## Installing OpenCV from prebuilt binaries

1. Below Python packages are to be downloaded and installed to their default locations.

1.1. [Python-2.7.x](http://python.org/ftp/python/2.7.5/python-2.7.5.msi).

1.2. [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download).

1.3*. [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe) (*Matplotlib is optional, but recommended since we use it a lot in our tutorials*).*

1. Install all packages into their default locations. Python will be installed to **C:/Python27/**.

11

1. After installation, open Python IDLE. Enter import numpy and make sure Numpy is working fine.
2. Download latest OpenCV release from [sourceforge site](http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.6/OpenCV-2.4.6.0.exe/download) and double-click to extract it.
3. Goto **opencv/build/python/2.7** folder.
4. Copy **cv2.pyd** to **C:/Python27/lib/site-packeges**.
5. Open Python IDLE and type following codes in Python terminal.
6. >>> import cv2
7. >>> print cv2.\_\_version\_\_

If the results are printed out without any errors, congratulations !!! You have installed OpenCV-Python successfully.

**4.2.5.3 Deep Neural Network Module**

4.2.5.3.1 BLOB:

BLOB stands for Binary Large OBject and refers to a group of connected pixels in a binary image. The term "Large" indicates that only objects of a certain size are of interest and that the other "small" binary objects are usually noise. There are three procedures regarding BLOB analysis.

1. BLOB extraction
2. BLOB representation
3. BLOB classification

1.BLOB extraction

The purpose here is to isolate the BLOBs (objects) in a binary image. As mentioned above, a BLOB consists of a group of connected pixels. Whether or not two pixels are connected is defined by the connectivity, that is, which pixels are neighbours or not. There are two main types of connectivity. The 8-connectivity and the 4-connectivity. The 8-connectivity is better than the 4-connectivity, but the 4-connectivity is often applied since it requires fewer computations, hence you process your image faster.

Moreover, a number of different algorithms exist for finding the BLOBs and such algorithms are usually referred to as *connected component analysis* or *connected component labeling*. Such algorithms are the Recursive Grass-Fire Algorithm, the Sequential Grass-Fire Algorithm, etc (I will not explain here how they work, but you can search by yourself).

12

2.BLOB representation

When you have extracted your BLOB, the next step is now to classify the different BLOBs. For example we want to classify each BLOB as either a circle of not a circle, or a human vs. non-human BLOB. The classification process consists of two steps. First, each BLOB is represented by a number of characteristics, denoted as *features*, and second some matching method is applied to compare the features of each BLOB with the features of the type of object we are looking for. For example, to find circles we could calculate the circularity of each BLOB and compare that to the circularity of a perfect circle. So, BLOB representation is a matter of converting each BLOB into a few representative numbers. That is, keep the relevant information and ignore the rest. Such of information include for example, the area, the circularity, the compactness, the perimeter, the center of mass, etc...

3.BLOB classification

The task here is to determine which BLOB for example, is a circle and which not. The question here now is how to define which BLOBs are circles and which are not based on their features that we mentioned earlier. For this purpose usually you need to make a *prototype model* of the object you are looking for. That means, what are the feature values of a perfect circle and what kind of deviation will you accept. This prototype model could differ from a simple *box classifier*.

## **How does Blob detection work ?**

SimpleBlobDetector, as the name implies, is based on a rather simple algorithm described below. The algorithm is controlled by parameters ( shown in bold below )  and has the following steps. Scroll down to know how the parameters are set.

1. **Thresholding :** Convert the source images to several binary images by thresholding the source image with thresholds starting at **minThreshold**. These thresholds are incremented  by **thresholdStep**until **maxThreshold**. So the first threshold is **minThreshold,**the second is **minThreshold**+ **thresholdStep,**the third is **minThreshold**+ **2 x thresholdStep**, and so on.
2. **Grouping :**In each binary image,  connected white pixels are grouped together.  Let’s call these binary blobs.
3. **Merging**  : The centers of the binary blobs in the binary images are computed, and  blobs located closer than **minDistBetweenBlobs**are merged.

**13**

**4.Center & Radius Calculation :** The centers and radii of the new merged blobs are computed and returned.

## **Filtering Blobs by Color, Size and Shape**

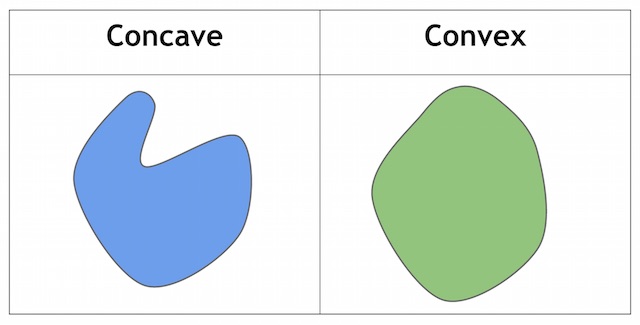
The parameters for SimpleBlobDetector can be set to filter the type of blobs we want.

* **By Color : [ Note : This feature appears to be broken. I checked the code, and it appears to have a logical error ]**First you need to set **filterByColor** = 1. Set **blobColor** = 0 to select darker blobs, and blobColor = 255 for lighter blobs.
* **By Size :**  You can filter the blobs based on size by setting the parameters **filterByArea = 1,**and appropriate values for **minArea** and **maxArea**. E.g.  setting **minArea** = 100 will filter out all the blobs that have less then 100 pixels.
* **By Shape :**Now shape has three different parameters.
  1. **Circularity :**This just measures how close to a circle the blob is. E.g. a regular hexagon has higher circularity than say a square. To filter by circularity, set**filterByCircularity** = 1.  Then set appropriate values for **minCircularity** and **maxCircularity.** Circularity is defined as  
       
     \frac{4*\pi*Area}{perimeter * perimeter}

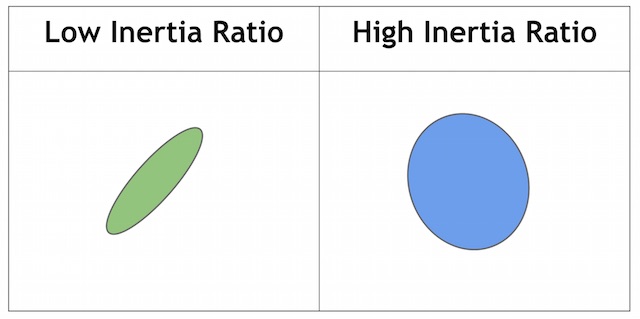
This means that a circle has a circularity of 1, circularity of a square is 0.785, and so on.

* 1. **Convexity :**A picture is worth a thousand words.  Convexity is defined as the (Area of the Blob / Area of it’s convex hull). Now, Convex Hull of a shape is the tightest convex shape that completely encloses the shape.  To filter by convexity, set **filterByConvexity**= 1**,**followed by setting 0 ≤ **minConvexity**≤ 1and **maxConvexity**(≤1)

14

[](https://www.learnopencv.com/wp-content/uploads/2015/02/concave-convex.jpg)

**3. Inertia Ratio :**Don’t let this scare you. Mathematicians often use confusing words to describe something very simple. All you have to know is that this measures how elongated a shape is. E.g. for a circle, this value is 1, for an ellipse it is between 0 and 1, and for a line it is 0. To filter by inertia ratio, set **filterByInertia =** 1**,**and set 0 **≤ minInertiaRatio**≤ 1and**maxInertiaRatio** (≤1)appropriately.

[](https://www.learnopencv.com/wp-content/uploads/2015/02/inertia.jpg)

15

* + 1. **Functional requirements**
       1. **Camera module**

Capturing the live video at the bus bay.

* + - 1. **Algorithm**

Fetching the video from the camera.

Detection of the bus accurately.

Estimating the total time of the bus near the bus bay.

**5 Software Design**

* 1. **Modules**

The software consists of the algorithm that uses single shot detection algorithm to detect the bus at the bus bay. It then starts and ends the timer to calculate the total time the bus has been stopped at the bus bay and if the bus doesn’t stop for a minimum specified time at the stop then the feedback is passed on to higher authorities.

**16**

**5.2 Use case Diagram**

**A close up of text on a white background

Description generated with high confidence**

Fig: Use case Diagram

The system consists of two users Agent, Administrator. The Agent is subdivided into camera and Algorithm where the camera module is used to capture video from the bus stop and the algorithm receives the video input from the camera module and detects the bus from the video and then calculates the in-time and out-time of the bus at the bus bay

**17**

**5.3 Class diagram**

A screenshot of a cell phone

Description generated with high confidence

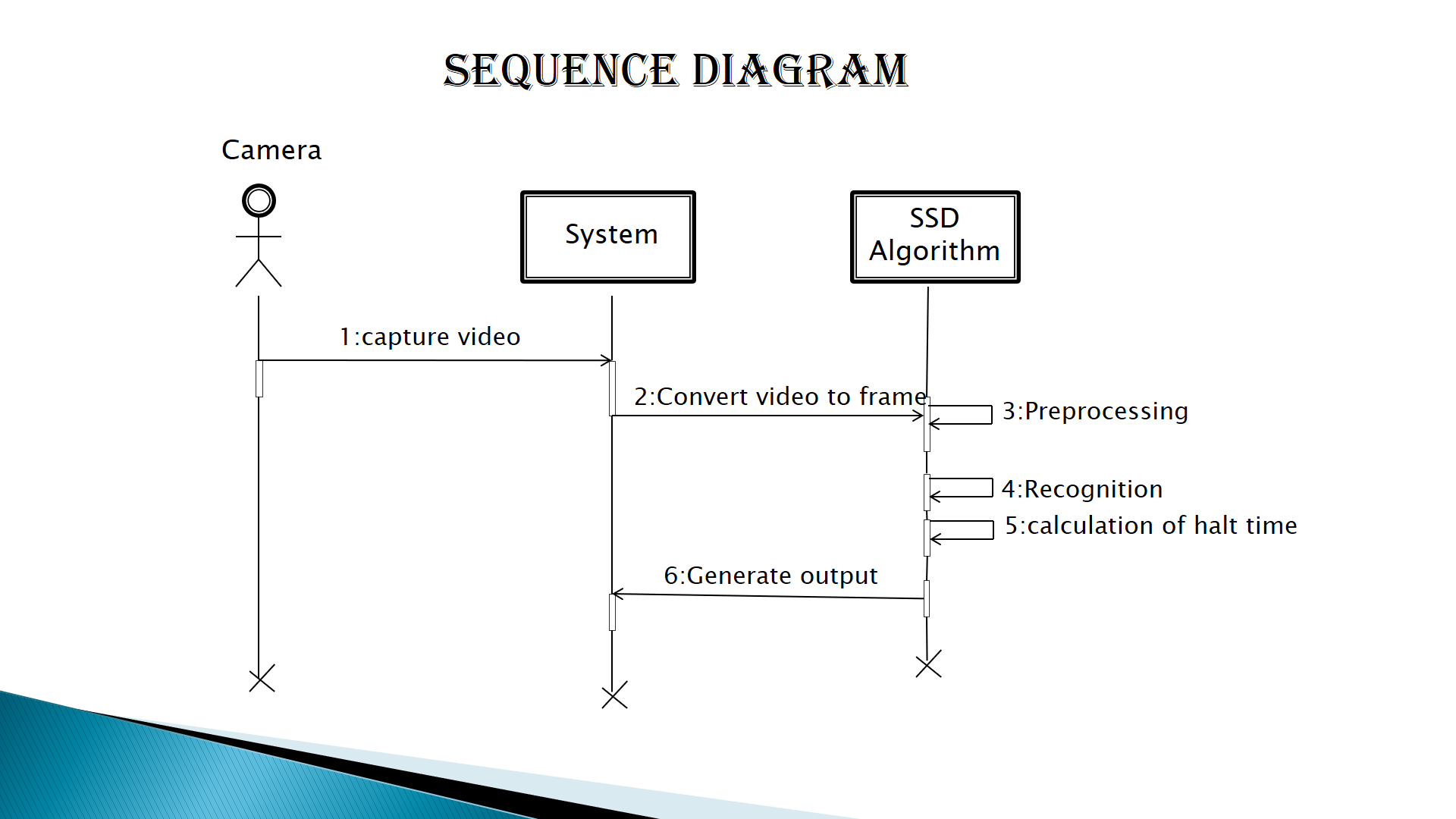
Fig: Class Diagram

As shown in fig, there are three main classes capture images, InTime and outTime, pre processing.

The pre processing class is an aggregation of recognition, segmentation, extraction, detection class. The detection class consists of a single function plate\_region which locates the bus and then in the recognition class we recognise the bus after the extraction and segmentation.

**18**

**5.4 Sequence diagram**



**Fig: Sequence Diagram**

The camera module records the video and the algorithm processes the video for the detection of bus from the video and then by calculating the in time and out time producing the time of bus near the bus bay.

19

**5.5 Activity diagram**

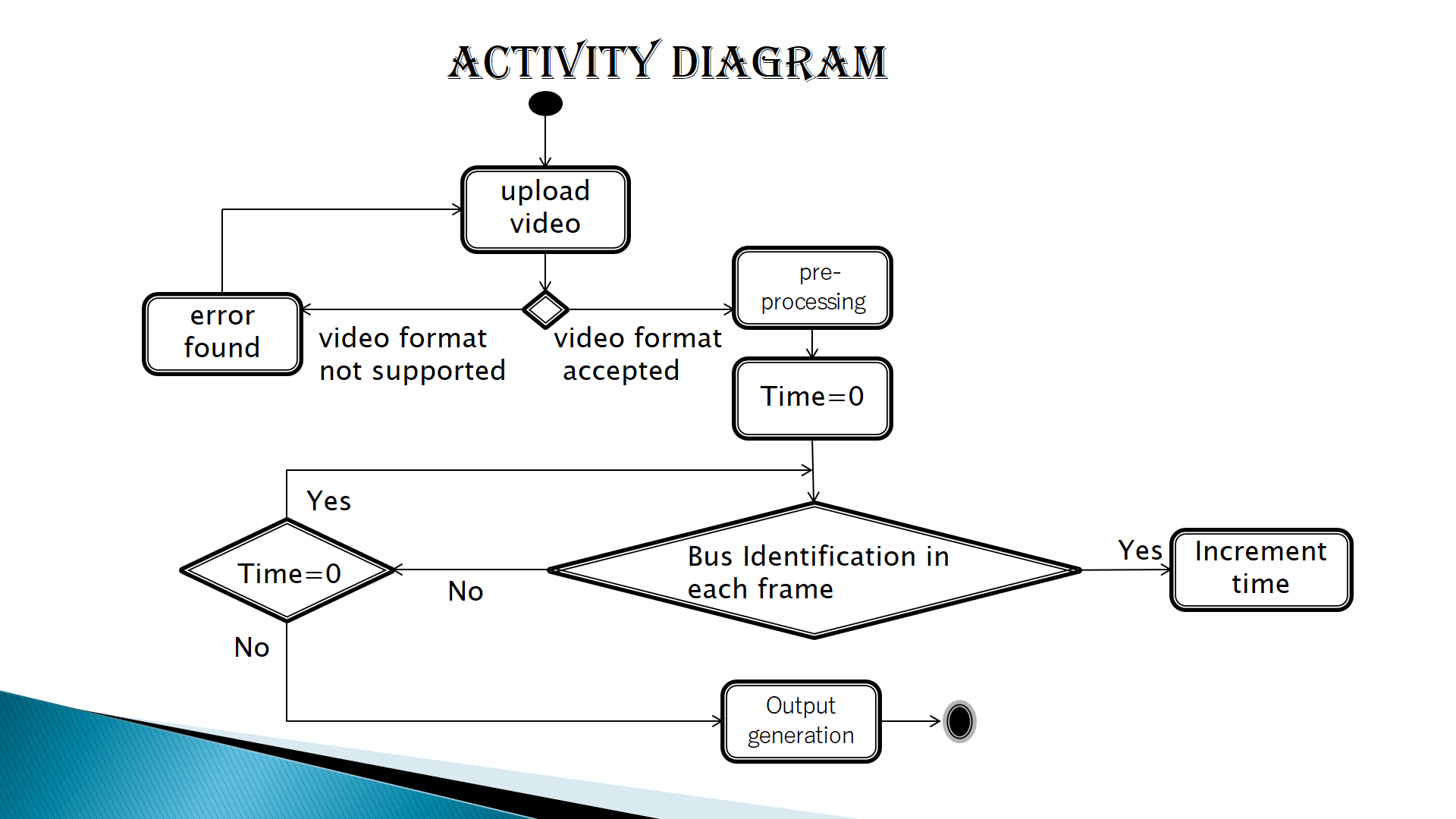
****

Fig: Activity Diagram

The algorithm fetches the video from the camera module and it uses single shot detection algorithm to detect the bus when it arrives at bus bay. This recognition is done using the trained caffe model used in the algorithm. When the bus is detected, the timer is set to on and each frame is checked for bus until it disappears when it disappears it breaks and it calculates the total time the bus is present in the frame which gives the total time that the bus has waited at bus bay for for safe boarding of passengers and according to that the feedback is given to the authorities.

**20**

**5.6 Work Flow**

A picture containing screenshot

Description generated with very high confidence

Fig: Work Flow

First the camera captures the live video stream at the bus bay. The algorithm fetches the video from the camera module and it uses single shot detection algorithm to detect the bus when it arrives at bus bay. This recognition is done using the trained caffe model used in the algorithm. When the bus is detected, the timer is set to on and each frame is checked for bus until it disappears when it disappears it breaks and it calculates the total time the bus is present in the frame which gives the total time that the bus has waited at bus bay for for safe boarding of passengers and according to that the feedback is given to the authorities.

**21**

**6 Implementation**

The implementation process is a bit complex since different kind of technologies must integrated in a manner that will allow them to work in a synchronized fashion without causing any errors. There are 4 main aspects we need to address

1 Environment in which the system runs.

2 Model development and training.

3 Deployment of the system.

**6.1 Environment**

The project mainly uses python language and its packages for everything related to machine learning and pre-processing. To properly manage the packages installed we use anaconda navigator software to create and install all the necessary packages. This environment is deployed on the server where the model resides to run the python scripts containing the machine learning code. The environment consists of the following important packages

1 numpy

2 Imutils

3 scipy

4 matplotlib

5 skimage

6 deep neural networks

7 Opencv

We use python programming for our system building. In python we import the Opencv library and then we are using this for image processing with the above packages. We detect the bus when the bus arrives at the bus bay and the time at bus bay is calculated.

**6.2 Model development and training**

Model development is nothing but teaching a black box containing neural networks the data

22

we have. That is we are trying to make it identify unique features and differences that classify or predict a bus is present in the frame or not.

To achieve this, we are using convolutional neural networks algorithm consisting of various layers and different functions performing some mathematical operations on the input data and its labels provided.

Before going to the model which we developed we need to understand some functions like convolution, maxpooling, activation, flatten, dense and dropout which form the neural network.

**6.2.1 Neural networks**

**6.2.1.1 Convolution**

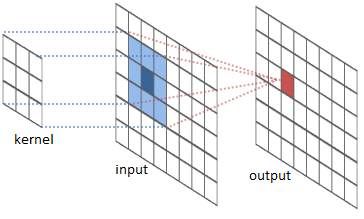


Fig: convolution

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not

23

practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation

**6.2.1.2 Maxpooling**

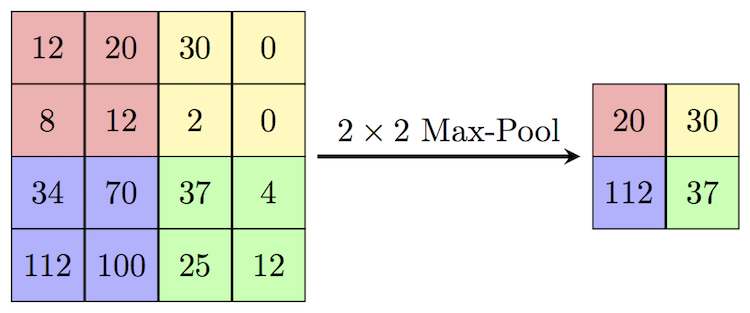


Fig: Maxpooling

24

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation.

**6.2.1.3 Activation**

In artificial neural networks, the activation function of a node defines the output of that node, or "neuron," given an input or set of inputs. This output is then used as input for the next node and so on until a desired solution to the original problem is found. 

**6.2.1.4 Flatten**

It flattens the input by reducing it to lower dimensions.

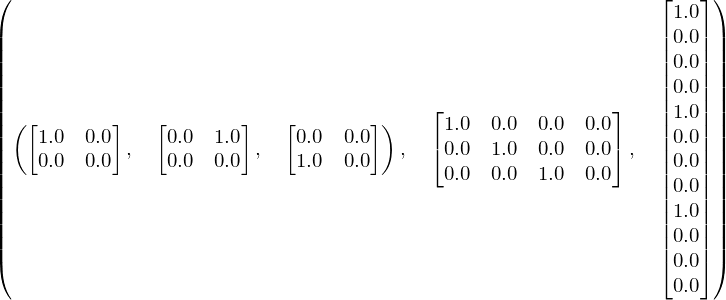


Fig: flatten

**25**

**6.2.1.5 Dense**

A dense layer represents a matrix vector multiplication. The values in the matrix are the trainable parameters which get updated during backpropagation. So you get a m dimensional vector as output. A dense layer thus is used to change the dimensions of your vector. Mathematically speaking, it applies a rotation, scaling, translation transform to your vector.

**6.2.1.6 Dropout**

A dropout layer is used for regularization where you randomly set some of the dimensions of your input vector to be zero with probability given. A dropout layer does not have any trainable parameters i.e. nothing gets updated during backward pass of backpropagation.Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

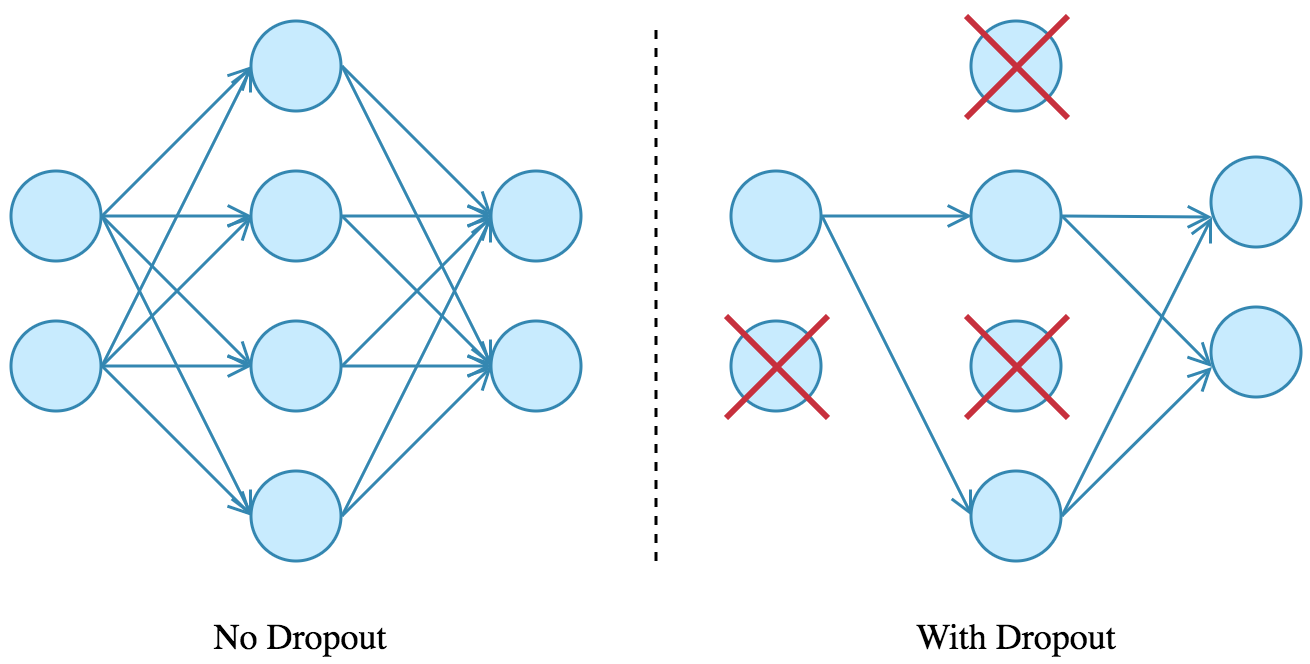


Fig: dropout

26

**6.2.1.7 Convolutional neural network with all connected layers**

Computers read images as pixels and it is expressed as matrix (NxNx3) — (height by width by depth). Images makes use of three channels (rgb), so that is why we have a depth of 3. The Convolutional Layer makes use of a set of learnable filters. A filter is used to detect the presence of specific features or patterns present in the original image (input). It is usually expressed as a matrix (MxMx3), with a smaller dimension but the same depth as the input file. This filter is convolved (slided) across the width and height of the input file, and a dot product is computed to give an activation map. Different filters which detect different features are convolved on the input file and a set of activation maps is outputted which is passed to the next layer in the CNN. See fig 6.9. In our case we are using a CNN that contains 25 million weights each of which are continuously being modified when data of 200 patients flows through them. The input dimension on our neural network is Nx64x64x64xP where N is the total no of samples of training data and 64x64x64 is the dimension of the pre-processed data of a single patient and 1 is the total number of such sets that the network is going to take as input. In our case N=200 and P=1. In our case the output is only a single probability hence there is only one node in the output layer. See fig 6.10 to better understand the internal layers.

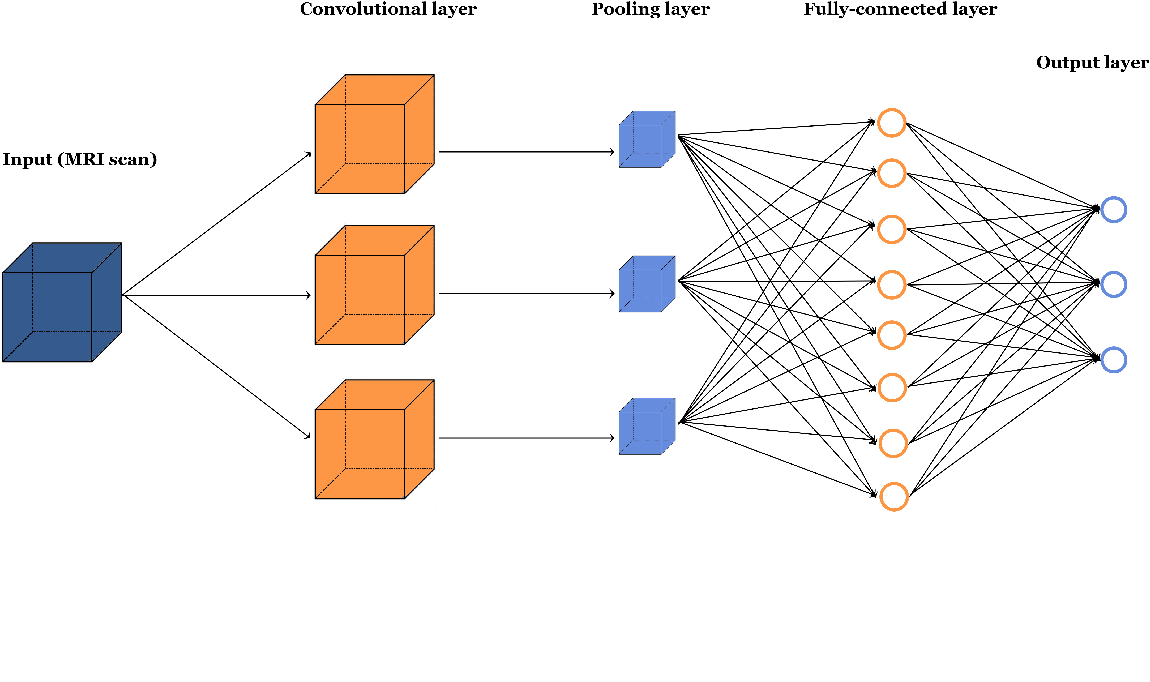


Fig: 3d CNN

27

**6.3 Trained Model**

We have used caffe model for our system. The caffe model is trained for detecting the bus using the pictures that we used in the model for training. The model takes the features from the pictures and predicts the required output.

**Time module**

Python has defined a[module](https://www.geeksforgeeks.org/python-modules/), “time” which allows us to handle various operations regarding time, its conversions and representations, which find its use in various applications in life. The beginning of time is started measuring from **1 January, 12:00 am, 1970** and this very time is termed as “**epoch**” in Python.

Operations on time module:

**time()** :- This function is used to count the number of **seconds elapsed since the epoch.**

import time

 # using time() to display time since epoch

print ("Seconds elapsed since the epoch are : ",end="")

print (time.time())

Output:

Seconds elapsed since the epoch are : 1470121951.9536893

**6.4 Prediction**

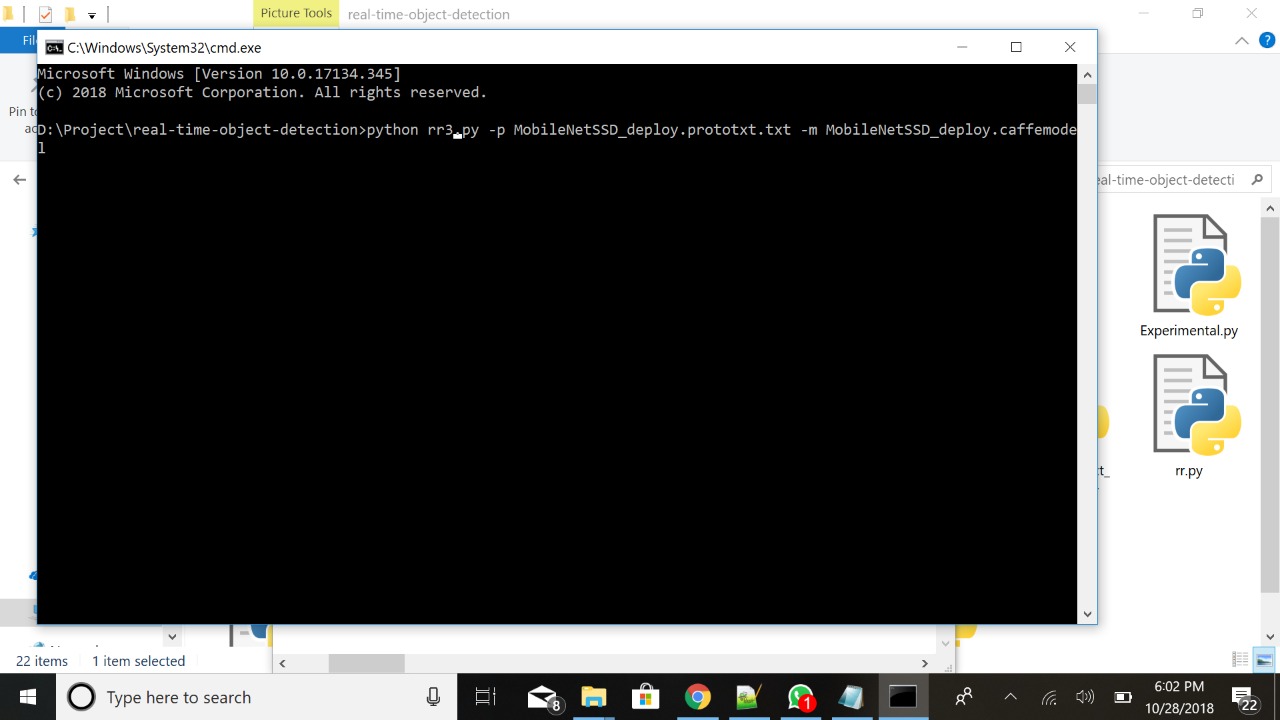
The above trained model predicts whether the bus is present in the video frame or not, if the system detects the bus in the frame it draws the bounding box with the obtained coordinates and prints the confidence with which the system is able to detect the bus which indicates the

28

user that a bus is detected in the video frame. We have set the minimum confidence for detection of objects as 50%. If the object confidence is greater than 50% the object is detected.

**7 Testing**

The model we developed was trained on 1000 data samples which had labels. This had made the model to detect the object accurately most of the times. We had tested our model using three test cases. In these three test cases the object is detected only when the confidence is greater than 50%.



29

The first testcase is the picture of an car which our system did not detect it as bus which is a success.

A screenshot of a computer screen

Description generated with very high confidence

30

The second test case is the picture of an bus which our system has detected as bus with confidence of 98%.

A screenshot of a computer screen

Description generated with very high confidence

31

The third test case is the picture of an car along with the bus here our system detected the bus with a confidence of 90%.

A screenshot of a computer screen

Description generated with very high confidence

**32**

**8 Results**

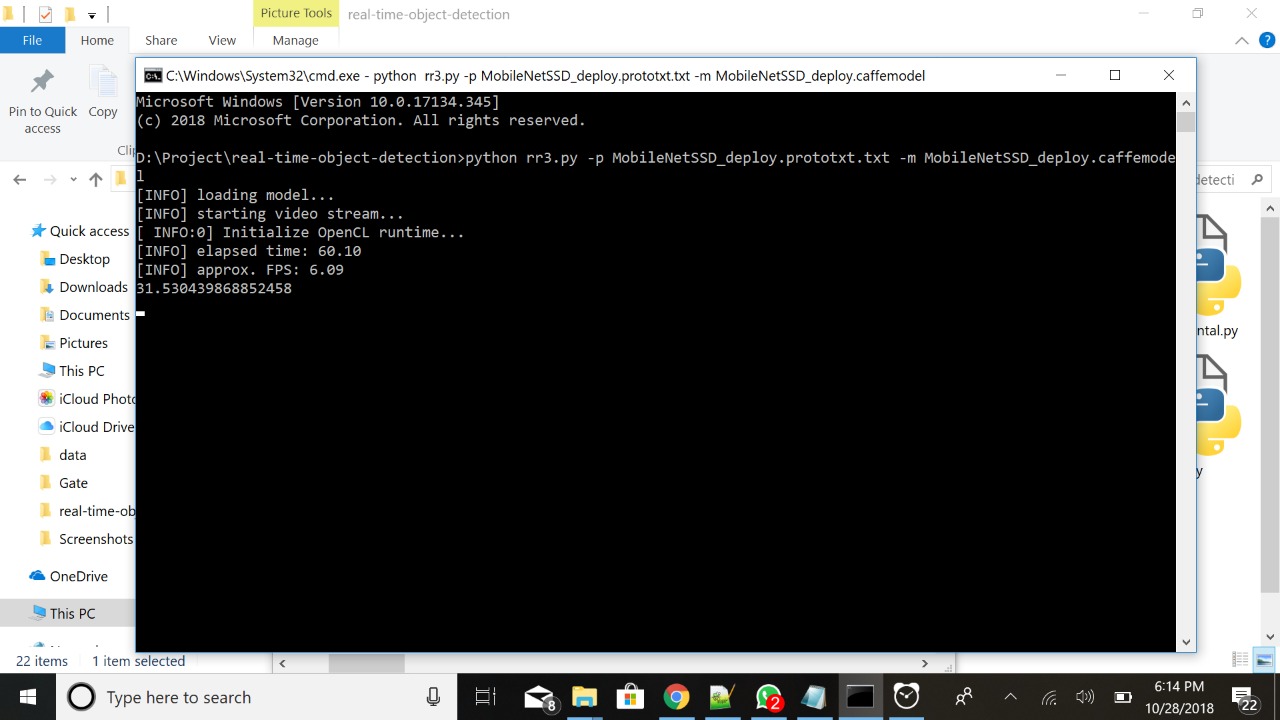
When the model was run on the test data the results were as follows.

The accuracy of the model is around 90%

A screenshot of a computer screen

Description generated with very high confidence

**33**

****

**8.1 Accuracy**

The overall system accuracy was found to be at 85% . That is out of 100 predictions 85 of them are accurate and there is a lot of scope for improvement.

Further analysis was done using contingency matrix.

**9 Code**

**# import the necessary packages**

**from imutils.video import VideoStream**

**from imutils.video import FPS**

**import numpy as np**

**34**

**import argparse**

**import imutils**

**import time**

**import cv2**

**# construct the argument parse and parse the arguments**

**ap = argparse.ArgumentParser()**

**ap.add\_argument("-p", "--prototxt", required=True, help="path to Caffe 'deploy' prototxt file")**

**ap.add\_argument("-m", "--model", required=True,help="path to Caffe pre-trained model")**

**ap.add\_argument("-c", "--confidence", type=float, default=0.2,help="minimum probability to filter weak detections")**

**args = vars(ap.parse\_args())**

**# initialize the list of class labels MobileNet SSD was trained to**

**# detect, then generate a set of bounding box colors for each class**

**CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "diningtable","dog", "horse", "motorbike", "person", "pottedplant", "sheep","sofa", "train", "tvmonitor"]**

**COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))**

**# load our serialized model from disk**

**print("[INFO] loading model...")**

**35**

**net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])**

**# initialize the video stream, allow the cammera sensor to warmup,**

**# and initialize the FPS counter**

**print("[INFO] starting video stream...")**

**vs = VideoStream(src=0).start()**

**time.sleep(2.0)**

**fps = FPS().start()**

**flag=False**

**tt=0**

**c=0**

**# loop over the frames from the video stream**

**while True:**

**# grab the frame from the threaded video stream and resize it # to have a maximum width of 400 pixels**

**frame = vs.read()**

**frame = imutils.resize(frame, width=400)**

**# grab the frame dimensions and convert it to a blob**

**(h, w) = frame.shape[:2]**

**blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)**

**36**

**# pass the blob through the network and obtain the detections and predictions**

**net.setInput(blob)**

**detections = net.forward()**

**# loop over the detections**

**for i in np.arange(0, detections.shape[2]):**

**# extract the confidence (i.e., probability) associated with the prediction**

**confidence = detections[0, 0, i, 2]**

**# filter out weak detections by ensuring the `confidence` is # greater than the minimum confidence**

**if confidence > args["confidence"]:**

**# extract the index of the class label from the**

**# `detections`, then compute the (x, y)-coordinates of**

**# the bounding box for the object**

**idx = int(detections[0, 0, i, 1])**

**box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])(startX, startY, endX, endY) = box.astype("int")**

**# draw the prediction on the frame**

**if CLASSES[idx]=="bus":**

**c=c+1**

**label = "{}: {:.2f}%".format(CLASSES[idx],confidence \* 100)**

**37**

**cv2.rectangle(frame, (startX, startY), (endX, endY),COLORS[idx], 2)**

**y = startY - 15 if startY - 15 > 15 else startY + 15**

**cv2.putText(frame, label, (startX, y),**

**cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, COLORS[idx], 2)**

**if c==1:**

**print("Bus Detected")**

**if flag==False:**

**st=time.time()**

**flag=True**

**else:**

**if flag==True:**

**et=time.time()**

**tt=tt+(et-st)**

**flag=False**

**# show the output frame**

**cv2.imshow("Frame", frame)**

**key = cv2.waitKey(1) & 0xFF**

**# if the `q` key was pressed, break from the loop**

**38**

**if key == ord("q"):**

**break**

**# update the FPS counter**

**fps.update()**

**# stop the timer and display FPS information**

**fps.stop()**

**print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))**

**print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))**

**# do a bit of cleanup**

**cv2.destroyAllWindows()**

**print("Total time bus present (Time in sec): ")**

**print(c/fps.fps())**

**if (c/fps.fps()>30):**

**print("Passangers boarded safely")**

**else:**

**print("Demerit for driver")**

**vs.stop()**

**cv2.waitKey(0)**

**39**

**10 Conclusion**

The system which we developed has an accuracy of about 85 percent and is really good at identifying bus at the bus bay from which we can calculate the total waiting time of the bus at the bus bay. If the bus doesn’t stop for atleast 30seconds then the driver will be given an demerit feedback and if the bus waits it is considered that all the passengers have boarded the bus safely.

**11 Future Scope**

* Video pre-processing during conditions of low-light to increase the performance of the system under extreme weather conditions.
* Training the system to identify number plate effectively despite the present of noise.
* This training helps to improve the performance and there by producing accurate results.
* Designing an user interface to provide the feedback for the driver and on how much time the haltime of bus should be increased at a bus stop**.**

**12 References**

<https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv>.

https://www.ijera.com/special\_issue/ICIAC\_April\_2014/EN/V7/EN-2604549.pdf

40

<http://academicscience.co.in/admin/resources/project/paper/f201706271498554084.pdf>

<http://www.kscst.iisc.ernet.in/sppArchive/public/Abstract/038/8011.pdf>

<http://www.ijemr.net/DOC/BusIdentificationModuleForVisuallyImpaired(452-455).pdf>

41

PHOTOGRAPH

