Density based traffic monitoring system

Kadam Anuja,Karkhile Siddhi, Chaudhari Kanchan, Bhagwat Divya

How to Cite this Article:

Anuja, K., Siddhi, K., Kanchan, C. & Divya, B. (2026). Density based traffic monitoring system. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.227

https://doi.org/10.55041/ijcope.v2i4.227

ABSTRACT

The Density Based Traffic Monitoring System through IoT using ResNet is an intelligent framework designed to optimize traffic management and reduce congestion in urban areas through automated analysis of vehicular flow. Traditional traffic control systems rely on fixed signal timing, which often leads to unnecessary delays and traffic jams, especially during peak hours. To overcome these limitations, this system integrates Internet of Things (IoT) technology with deep learning to create a dynamic and adaptive traffic control mechanism.

IoT-enabled surveillance cameras are installed at various intersections to continuously capture live video feeds of the traffic. These video streams are transmitted to a processing unit, where image frames are extracted and analysed using the ResNet (Residual Network) model. ResNet, known for its high accuracy and efficient feature extraction capabilities, is used to detect and classify vehicles in each lane. By counting the number of vehicles, the system determines the traffic density in real time. Based on this density information, the system automatically adjusts the traffic signal duration to allow smoother vehicle movement in congested lanes, thus reducing waiting times and improving overall traffic flow.

Additionally, the data collected from the IoT sensors and cameras can be stored and analysed for long-term traffic pattern prediction, road planning, and infrastructure management. The integration of deep learning and IoT enables continuous monitoring without human intervention and ensures real-time responsiveness to changing traffic conditions. This system contributes to the development of smart cities by enhancing traffic efficiency, reducing fuel consumption, minimizing carbon emissions, and improving commuter safety and convenience.

Keywords: IoT, ResNet, Traffic Monitoring, Vehicle Detection, Traffic Density Estimation, Smart City, Deep Learning

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Density Based Traffic Monitoring System through IoT using ResNet is a modern and intelligent solution developed to manage and optimize traffic flow efficiently in growing urban environments. As cities continue to expand and the number of vehicles on roads increases rapidly, managing traffic congestion has become one of the major challenges for authorities. Conventional traffic management systems rely on fixed time-based signal operations, which do not account for real-time traffic variations. These static systems often cause traffic jams at busy intersections, excessive waiting times for vehicles, unnecessary fuel consumption, and increased air pollution. Therefore, there is a strong need for a smart and adaptive system that can monitor, analyse, and control traffic conditions dynamically based on real-time vehicle density.

The proposed system integrates the Internet of Things (IoT) with advanced deep learning techniques to create a fully automated and intelligent traffic management framework. IoT-enabled surveillance cameras are installed at various junctions and road intersections to capture continuous live video feeds of the traffic movement. The captured video data is transmitted to a central processing unit, where individual frames are extracted and analyzed using the ResNet (Residual Network) deep learning model. ResNet, known for its superior feature extraction and image classification accuracy, is applied to detect and count the number of vehicles in each lane efficiently. By processing this information, the system estimates the current traffic density and evaluates which lane requires more or less signal duration.

Based on this real-time density analysis, the system dynamically adjusts traffic signal timings to provide longer green light durations to heavily congested lanes while minimizing the wait time for less crowded ones. This intelligent and adaptive signal control reduces unnecessary delays, prevents traffic bottlenecks, and ensures smooth vehicle movement across intersections. Additionally, by minimizing idle time at traffic lights, the system helps lower fuel consumption and carbon emissions, contributing to environmental sustainability and energy efficiency.

The IoT infrastructure ensures continuous communication and coordination between cameras, sensors, and the control unit. All the traffic data collected through IoT devices is transmitted to a centralized monitoring system, where it can be further analyzed for long-term traffic pattern prediction, city planning, and policy-making. The use of cloud-based data storage enables authorities to visualize traffic behaviour trends over time and take informed decisions for infrastructure development.

ResNet plays a crucial role in the accuracy and efficiency of the system. Its deep convolutional layers and skip connections allow the model to perform well even with large datasets and complex road conditions. This ensures that the system maintains high performance in different weather conditions, lighting variations, and diverse

environments. The combination of IoT for real-time data acquisition and ResNet for intelligent analysis makes the system highly reliable and scalable for smart city applications.

In addition to improving traffic flow, the system also enhances road safety by enabling authorities to monitor intersections in real time and detect irregular traffic conditions or violations. Alerts can be generated automatically in case of unexpected congestion, accidents, or equipment malfunctions. Such capabilities make the system not only efficient for everyday traffic management but also valuable for emergency response and urban safety management.

Overall, the Density Based Traffic Monitoring System through IoT using ResNet represents a significant advancement in intelligent transportation systems. It transforms traditional, rigid traffic control into a dynamic, data-driven, and responsive network capable of self-adjustment based on actual road conditions. By combining the computational strength of deep learning with the connectivity of IoT, the system offers an effective and sustainable solution for modern cities aiming to improve mobility, reduce pollution, and enhance the overall quality of life for citizens.

1.2 PROBLEM DEFINATION

The problem with existing traffic systems is their fixed signal timing, which fails to adapt to real-time traffic conditions, causing congestion and delays. The proposed system using IoT and ResNet aims to detect vehicle density automatically and adjust signal timing dynamically for efficient traffic flow and reduced waiting time.

CHAPTER 2
LITERATURE SURVEY

[1] Sachin Upadhye et al. proposed a new RTTFM-DL technique to detect vehicles, count vehicles, estimate speed, and determine traffic flow. The RTTFMDL technique involves a vehicle detection model using Faster RCNN with ResNet model. Moreover, a detection line based vehicle counting approach is designed, which is based on overlap ratio. Furthermore, the traffic flow is monitoring based on the estimated vehicle count and vehicle speed. For assuring the proficient results analysis of the RTTFM-DL technique, a sequence of experimental analyses is carried out and the results are inspected under distinct aspects. The experimental outcomes highlighted the betterment of the RTTFM-DL technique over the recent techniques. In future, the presented RTTFM-DL technique can be employed in real time scenarios such as logistics, healthcare, etc. The RTTFM-DL technique has gained improved outcomes with a higher accuracy of 0.975.

[2] S. Rafiq et al explain the existing system doesn't provide a transparent path for emergency vehicles during traffic congestion. Traffic Density Analysis, Ambulance, and Accident detection System Using Image Processing has been discussed in this proposed system. From the literature survey, we've found that RFID-based smart traffic control system provides an answer to the traffic congestion problem and this can be also an efficient method to supply a transparent path for the emergency vehicles when identified within the lane, as author also

implemented sharing of patient's vital data with hospital author updated Arduino uno with Arduino mega board so it'd be sufficient for storing of patient vital parameter and simultaneously it performs capturing of present status of traffic signal present in different path and author also added another system in the junction which repeatedly scans the density of the lanes so that the system can automatically allow the lane which has high density by this technique the emergency vehicles experience less congestion and reach faster to the destination and thus many life's were been saved.

[3] Patel, R et al introduced Emergency vehicle priority (EVP) systems could greatly reduce the traveling time for emergency vehicles like ambulances, which in turn could save many lives. The presented EVP used a neural network-based classifier to detect an ambulance siren. A mobile app was used to track the location of the ambulance. A decentralized network of IoT devices equipped with visual indicators was used to inform traffic of an incoming ambulance. Hence a makeshift emergency lane was set up on the roads. author's EVP system didn't require any modifications to be made to existing ambulances. Nodes could be added to the IoT network easily and without affecting the response speed of the system. As author's system used audio signals and WiFi communication to function, it was not affected by factors like rain, snow, fog, etc. author expect that author's proposed system enables ambulances and other emergency vehicles to reach their destinations as quickly as possible

[4] kherraki, Amine et al explain a comparative study using eight famous CNN architectures to classify emergency vehicle images. The design and implementation of an efficient deep learning system, have been carried out to automatically classify emergency and normal vehicles in traffic scenes. First, author did a preprocessing on the Vidhya Emergency Vehicle dataset to unify the image sizes. author have added three layers to each CNN architecture, in particular, "GlobalAveragePooling2D" layer to reduce the dimensions of the input image and accelerate the training, "Dropout" layer to avoid overfitting, and finally, "Dense" layer where author put the number of output classes. Later, author made simulations of each architecture, and author notice that DenseNet121 is the most appropriate model in real-time emergency vehicle classification with an accuracy of 95.14%, a F1 score of 93.87%, and an average order memory of 27.5 MB. Therefore, reached results are very promising and will definitely give an important added value to applications that will use author's best architecture for the classification of emergency vehicles.

[5] Sunil M et al proposed for saving a patient's life in a faster way possible. It is beneficial for users in case of emergencies as it saves time. With this Application, the ambulance can reach the patients as fast as possible using the video and detection algorithm. This system makes sure that more lives are saved . It is easier for detecting ambulance and low cost. The difficulty is if the ambulance is stuck after a distance of more than 300 metres then it would be very difficult to detect the ambulance.

Limitation\Future work- This system can be implemented to all other emergency vehicles in future like fire engine, police, cars etc., in future. author can share the patient information to the hospital in an easy way.

[6] Usaid, M et al describe the ambulance siren and traffic noises dataset has been developed in '.wav' formats in this research article. The collected audio signals are first preprocessed before extracting features using

Python's Librosa library. Twenty one Mel-frequency Cepstral Coefficient and the spectral centroid, roll of rate, spectral bandwidth, Chromastft, and zero-crossing rate are extracted. A GPU has been utilized for training an MLP model; during the training, the model achieved a 90 % accuracy. The work has focused on designing a dataset and feature extraction process, distinguishing ambulance sound from road noise being the most important. author haven't yet put the data to the test in a real-world environment. author plan to test the data in real time in the future by recording more audio datasets. The developed model can be used in practice after enhancing the dataset and testing in real-world scenarios. The trained model can be programmed on any microcontroller device and implemented in real-world applications to detect ambulance sirens and optimize the response time. In terms of complexity, the established MLP-model architect is simple enough to run on any micro-processing gadget or any board without a Graphical Processing Unit.

[7] Bhoomika G M et al., author were able to: Build a dataset so the software may need it for training and evaluation. The application used the design that was created in order to separate the ambulance from most other vehicles. While other traffic lights were turned red and author successfully got the computer to change that traffic light to green when it realised the captured vehicle was an ambulance and “a notification was sent to the traffic controller”. The ambulance, a vehicle used for emergencies, can travel rapidly and be at its destination in a matter of a few minutes thanks to the camera prediction and prevention. As once work activities of capture, training and detection have been accomplished, the simulation of the entire system may be completed successfully. The scope of research can be broadened to include all emergency vehicles in the coming.

[8] Mahmud, Umar et al explain traffic management in developing countries is a challenging issue. Traffic management systems as well as human resources are utilized to manage the traffic flow. Traffic congestion results in resource wastage in the form of fossil fuel, pollutes the urban environment, and creates annoying delays. These delays hamper quick EV transit through traffic intersection which can end in a fatal outcome. This work proposes a distributed A IoT-based EV transit system, built on Raspberry PI with minimal sensors including infrared sensors for detecting emergency lights and directional microphones for detecting sirens. This system makes decisions based on rules to ensure that an EV moves smoothly through crossings. Once an EV is detected on an incoming lane, the traffic signal is turned to green until the time the EV has successfully passed through the intersection. The departure lane is detected, and this information is shared with adjacent intersections to further reduce the transit time. As a result of the system, successful EV transit has increased from 25% to 62.5 percent

[9] Yarra Kavitha et al narrate the primary design objective of this project is to shorten the EV’s wait time at the intersection and to prevent accidents there, which can prevent the destruction of valuable property and the loss of lives. The system is totally autonomous in this case, with no driver interaction required to transmit SRM requests to the RSU. When compared to the delay caused by the GSM method, transmission and reception occur much more quickly, thanks to DSRC’s reliable communication. This method also avoids the usage of additional detectors and sensor circuits used in other approaches to determine the vehicle location or to detect the vehicle entering the pre-emption point, Hence the system is less expensive. A system of automated messages that may be utilized to inform the driver of the next open lane from an intersection that leads to the desired location may

also be incorporated. In densely crowded lo cations, decision support systems may establish communication to notify approaching intersections of the need for rapid action. If there is ever an emergency, this system can also be installed in non-emergency vehicles such as motorcycles that will assist the user in an emergency and this system may eventually be improved by creating a dynamic web or mobile application as an interface that allows for more intelligent automatic and manual control of the system.

[10] Dr. P. Sankar Babu et al describe the human labor of the road safety officer can be reduced by using digital control of traffic signs that is calibrated to the route's actual traffic volume. There is minimal need for human involvement because the system is fully automated. When a stolen car is detected, the light will turn red so that an officer can intervene, if one exists at the intersection. They will also receive text messages to help them be ready to stop the stolen car at the next intersection. When responding to an emergency, time is of the essence for vehicles such as ambulances and fire engines. They risk endangering many people's lives if they are stuck in traffic for long periods of time. As long as a rescue vehicle is waiting at the intersection after being given the all-clear, the light will turn green. The light will change to red when the emergency vehicle has passed through. The prototype may be improved upon by testing it with RFID readers that have a greater range. Additionally, GPS may be included into the stolen car detection module.

[11] Rawand Sulayman et al proposed a model that can detect emergency cars on a heavy traffic road. A populated region like Kurdistan faces too much traffic on the road and because of that emergency car like ambulance and fire-service fall into trouble middle of the road. author's model will solve this problem. It can be embedded with CCTV to track emergency can and give priority in that road to pass the emergency can. With this automated process, no human effort will be required to manually help such scenario. In author's project author used a customized Yolov5 object detection algorithm. YOLO an acronym for (You Only Look Once), it is an object detection algorithm that divides images into a grid system. Each cell within the grid is responsible for detecting objects within itself. YOLO models are used for Object detection with high performance which consists of 84 classes to detect and differentiate between 84 different objects. author's model is based on 4 classes which are (Firetrucks, Ambulance, Police Car, and Normal Cars) classes. author's model has achieved impressing results in detecting and identifying emergency cars of all kinds, for Police Cars author have the result of 98%, for Fire Trucks 96%, for Ambulances we've got 89% and for Normal Cars 97% results.

Limitation\Future Work - Probing deeper, the results in this research also provide a strong foundation for future work in awareness and in peripheral displays. author's area of future work is to feature Sound Recognition System for emergency vehicles which are recognized by their siren sound. The Siren Sound is a special signal sounded by alarm systems or emergency vehicles. When an emergency vehicle performs its task, the siren sound is issued to alert other drivers or pedestrians on the road.

[12] Santosh D. Pandure et al presents a cutting-edge solution for enhancing urban traffic management by prioritizing traffic signals for emergency vehicles. The integration of deep learning, multi-modal data analysis, and adaptive signal control promises to mitigate traffic congestion and optimize emergency response times.

Limitation\Future Scope: Future work will focus on real-world deployment and the fine-tuning of the system for broader urban implementation.

[13] Amrutasagar, K. et al., Implementation of the specified YOLOv7 ambulance detection model leads to the utilization of valid data about traffic in an urban setting, which further improves emergency services. The model's top performance as depicted by its F1 score of 0.90 at a confidence level of 0.444 and above, precision of 1.00 at a confidence of 0.816 and a mAP of 0.925 at IoU of 0.5 supports the fact that it identifies ambulances in traffic with little false alarms.

Limitation\Future scope - A success story in terms of ambulance detection for YOLOv7 has inspired possibilities and improvement in the utilization of the model in the future.

[14] Noor, A. et al presents the design and implementation of a novel ambulance detection system architecture to prioritize ambulance vehicles by turning the traffic light to green for saving patients' lives. The system is capable of handling multiple types of signal detection reports from different sources and formats by processing and storing them in the cloud. In particular, author exploit a YOLOv8 model to detect the ambulance and allow the traffic signals to turn green to speed up the ambulance's estimated arrival time at the hospital. author are the first to exploit a YOLOv8 model for the detection of ambulances, to the best of author's knowledge. To demonstrate the performance of author's YOLOv8 model, author collected and labeled 3000 images of ambulances from 10 different countries that cover various domains and languages. The dataset can benefit the research community in training and evaluating computer vision models for cross-lingual tasks. Moreover, author conducted 22 different experiments, 10 of which are pre-trained while the other 10 are non-pre-trained author conducted another two experiments, which author call the universal model for all of the countries that author have listed. Furthermore, author compared the performance of author's YOLOv8 model with other models presented in the literature including YOLOv5 and YOLOv7. The results of the experiments are quite promising where the universal model of YOLOv8 scored an average of 0.982, 0.976, 0.958, and 0.967 for the accuracy, precision, recall, and F1 score, respectively. The universal model scored the highest result among all countries because of the large amount of data for training where the model could understand the ambulance vehicles better.

CHAPTER 3

SCOPE OF THE PROJECT

Scope of Density Based Traffic Monitoring System through IOT is explained below

3.1 IoT-Based Smart Traffic Management System using IR Sensors

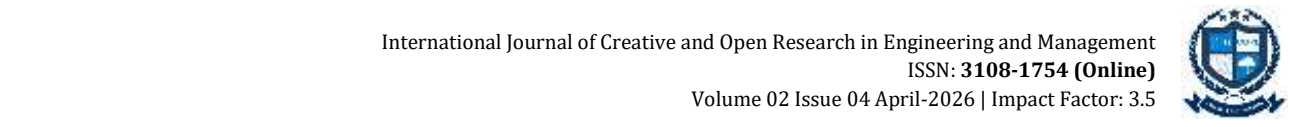
This existing system uses Infrared (IR) sensors to detect vehicle density at traffic junctions. The sensors are placed on roads to count vehicles and measure congestion levels in real time. The collected data is transmitted through IoT modules to a cloud server for processing and analysis. Based on traffic density, the system dynamically adjusts signal timings instead of using fixed timers. This helps reduce waiting time, fuel consumption, and traffic congestion in busy areas. Authorities can monitor traffic remotely using dashboards and take necessary actions. The system is cost-effective and easy to deploy in urban environments. It improves road efficiency and reduces manual traffic control. However, its performance may be affected by environmental conditions like dust or obstacles blocking sensors.

Reference URL : https://espjeta.org/jeta-v3i4p101/

3.2 GPS & Raspberry Pi Based Dynamic Traffic Density System

This system uses GPS technology and Raspberry Pi to monitor real-time traffic density and manage signals dynamically. Vehicles or traffic nodes send location and density data to a central system using IoT communication. The system analyzes traffic load and gives priority-based green signals, especially for emergency vehicles like ambulances. It reduces congestion, travel time, and harmful emissions by optimizing signal timing. The model supports real-time decision-making and automation without human intervention. It also improves road safety and minimizes accidents caused by unmanaged traffic. The system is scalable and can be extended with AI for better predictions. However, it depends on accurate GPS data and network connectivity. Overall, it is an efficient step toward smart city traffic management.

Reference URL: https://link.springer.com/chapter/10.1007/978-3-031-16364-7_22/



CHAPTER 4
METHODOLOGY

The methodology for Density Based Traffic Monitoring System through IOT is developed under waterfall model architecture as shown in the below figure 1.

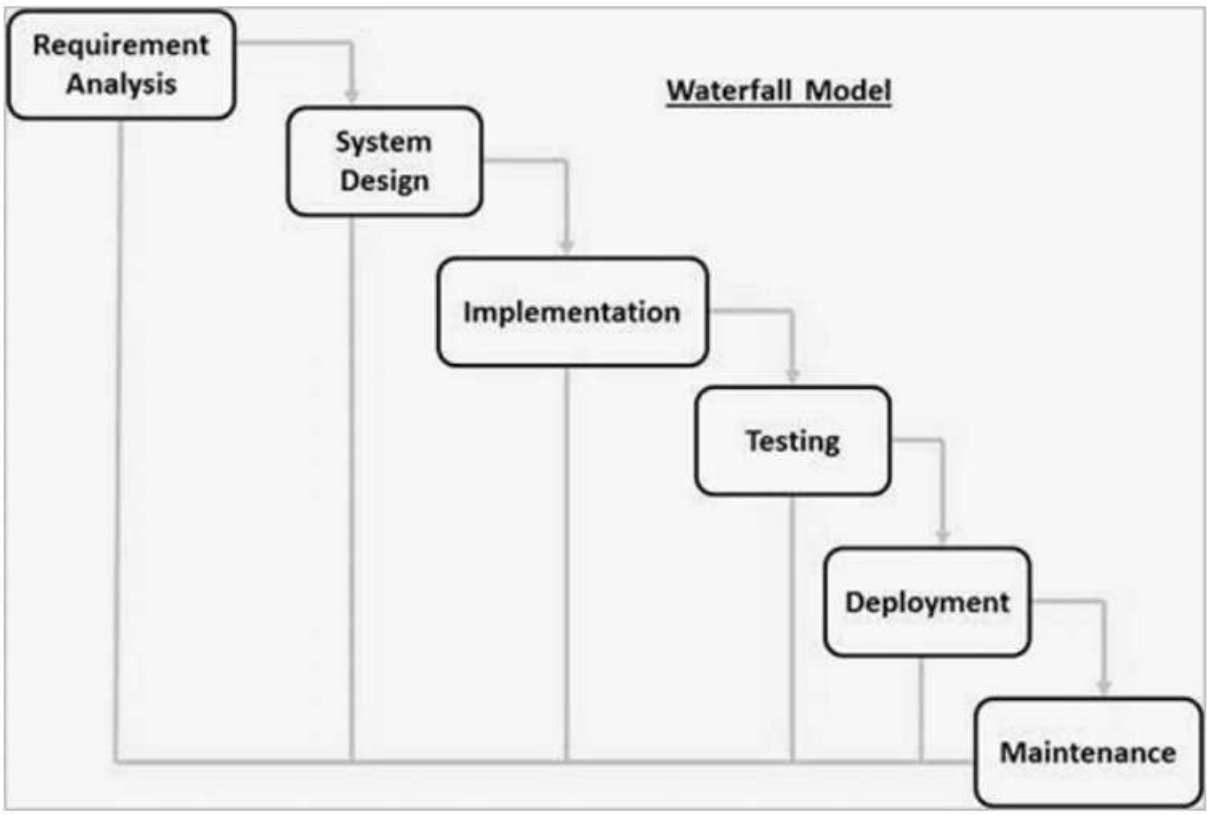


Fig 1 : Water fall model Architecture

The sequence phases in water fall model according to our project are mentioned below.

4.1 Requirement Analysis – Here requirement analysis are done based on following points

✓ Base paper for Density Based Traffic Monitoring System through IOT

4.2 System Design: The System of Density Based Traffic Monitoring System through IOT is designed by using the following hardware and software

Minimum Hardware Specification:

- CPU : core i5
- RAM : 8 GB
- HDD : 500 GB
- Micro Controller : Arduino UNO
- Camera : 48 MP
- Battery : 18650 limh battery
- Motor : DC Motor

Software Specification:

- Coding Language : Python
- Development Kit : SDK
- Front End : Tkinter
- Development IDE : Spyder, Arduino IDE
- Database : MySQL 5.5 Server

4.3 Implementation:

Proposed system is designed by using the following modules

- **Module A: Preprocessing**
  - **Input:** Frames captured from live traffic video streams.
  - **Process:** Extracted frames are resized, filtered, and enhanced to remove noise and improve image quality for accurate vehicle detection using the ResNet model.
  - **Output:** Preprocessed and high-quality image frames ready for normalization and model training.
- **Module B: Image Normalization**
  - **Input:** Preprocessed image frames from the traffic video stream.
  - **Process:** Pixel values of the images are scaled and normalized to a fixed range to maintain consistency and improve ResNet model performance during training and detection.
  - **Output:** Normalized image dataset suitable for accurate feature extraction and vehicle detection.
- **Module C: ResNet**
  - Input: Normalized image dataset containing traffic frames.
  - Process: The ResNet deep learning model performs feature extraction, vehicle detection, and classification using its residual learning architecture to ensure high accuracy and reduced training error.
  - Output: Trained model capable of identifying and analyzing traffic density for decision-making and signal management.
- **Module D: Decision Making**
  - **Input**: Vehicle count and density from ResNet output.
  - **Process:** Analyzes lane-wise density to decide signal timing.
  - **Output:** Optimized signal control for smooth traffic flow.

4.5 Deployment of the system:

The developed software is deployed in the laptop of above mentioned configuration with the help of the mentioned software.

4.6 Maintenance of the system:

As this software is tested for the quick recovery, so maintenance of the system is not a challenging task. This is because the tools and the software used are open source, so there is no question of licensing the required software.

CHAPTER 5

DETAILS OF DESIGN, WORKING AND PROCESSES

5.1 DETAILS OF DESIGN

5.1.1 Data Flow Diagrams

5.1.1.1 DFD level 0

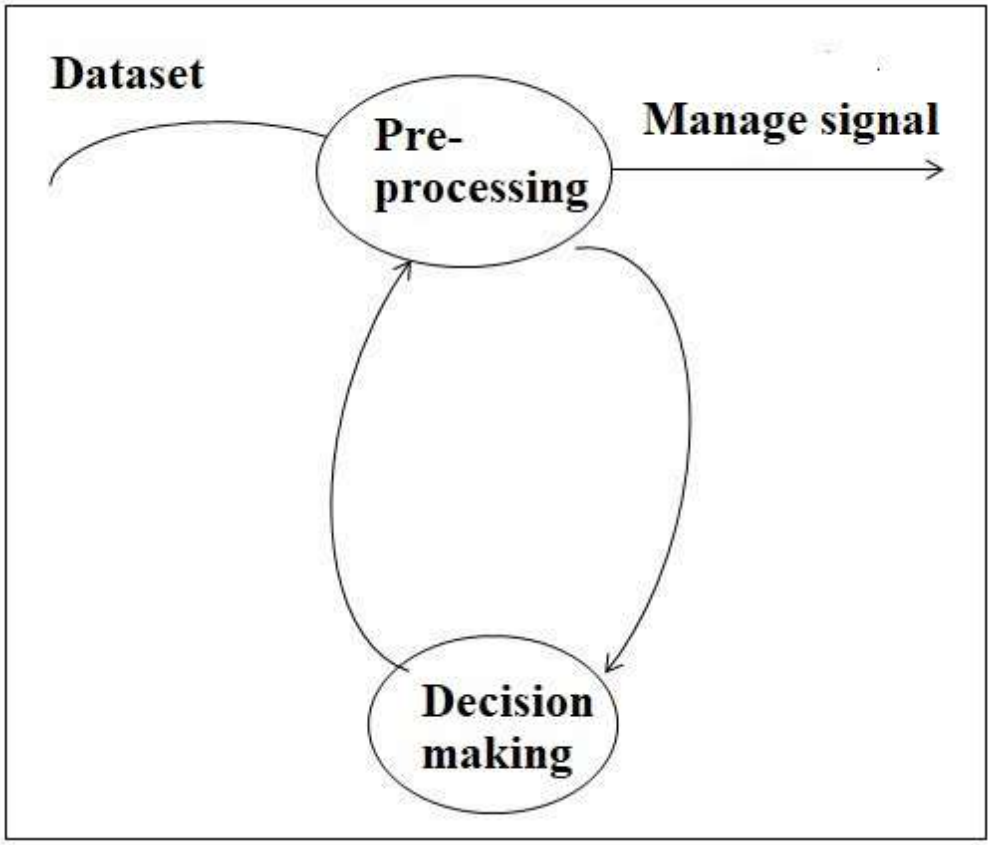


Fig 2 DFD level 0

The DFD 0 diagram for the data flow diagrams describes the flow of the approach. The DFD diagram provides the simplest flow wherein the dataset is provided to preprocessing and then decision making is applied to get alert manage signal.

5.1.1.2 DFD level 1

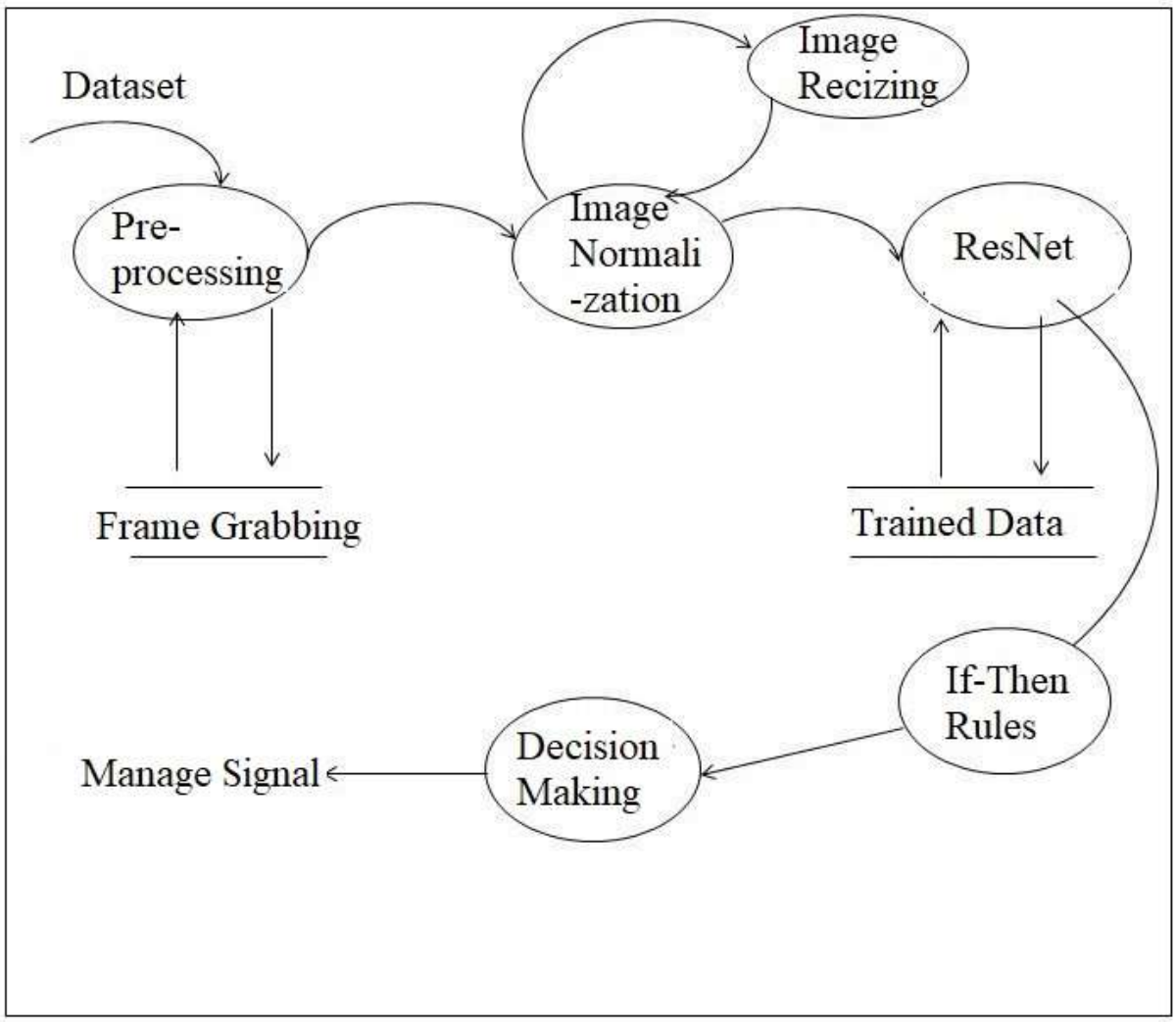
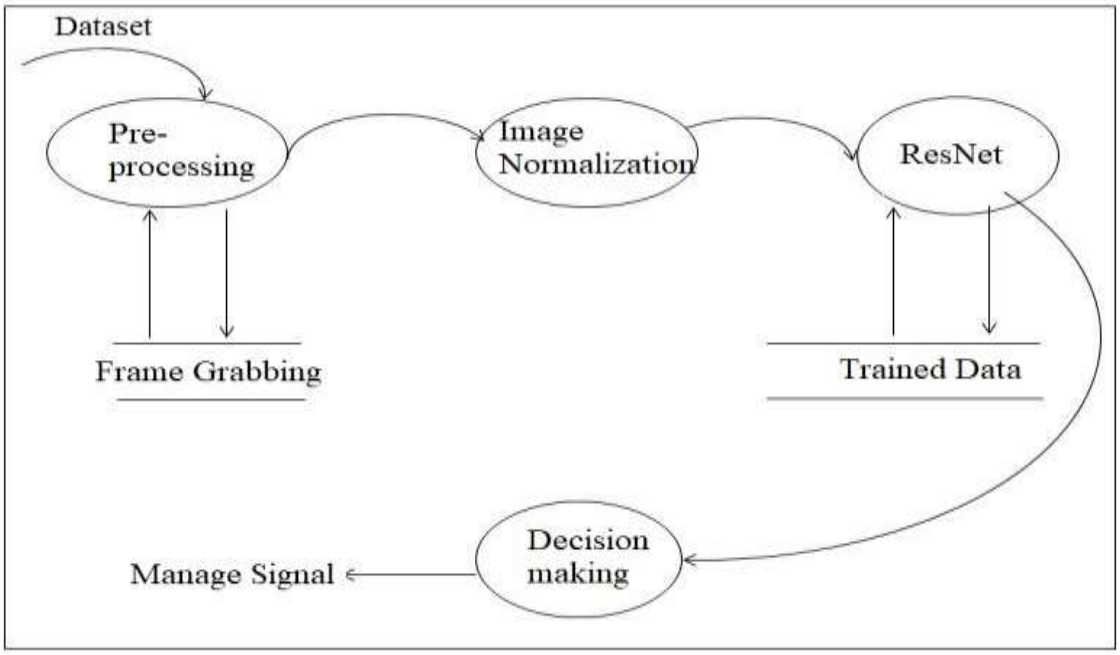


Fig 3 DFD level 1

The DFD 1 diagram provides even more details wherein the dataset is provided to preprocessing and then frame grabbing is done. After which preprocessing result is provided to image normalization, then ResNet is applied to get trained data which implements the decision making to get manage signal.



5.1.1.3 DFD level 2

Fig 4 DFD level 2

The DFD 2 diagram is the most detailed wherein the dataset is provided to preprocessing and then frame grabbing is done. After which preprocessing result is provided to image normalization and image resizing is done, then ResNet is applied to get trained data after which if-then rules is performed which implements the decision making to get manage signal.

5.1.2Activity Diagram

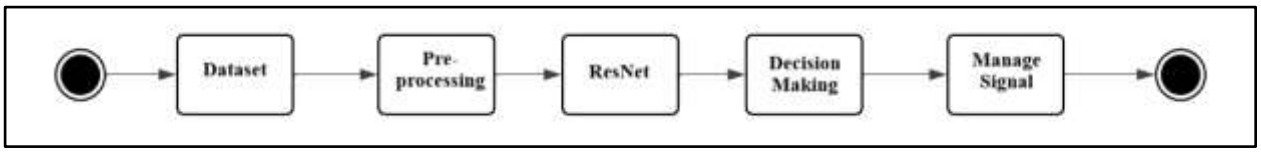
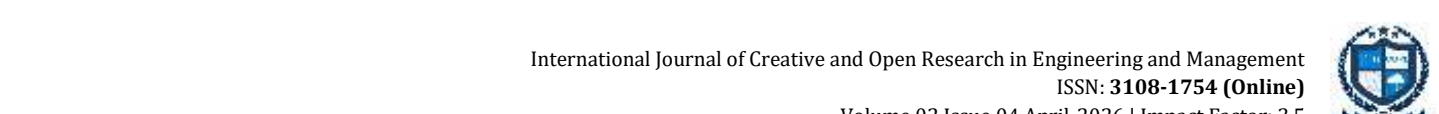


Fig 5.1 Activity Diagram

The Activity diagram lists the various activities performed in the proposed system wherein the start state is initiated and the dataset, preprocessing, image normalization, decision making, and manage signal and the system reaches the stop state.



5.1.3 Usecase Diagram

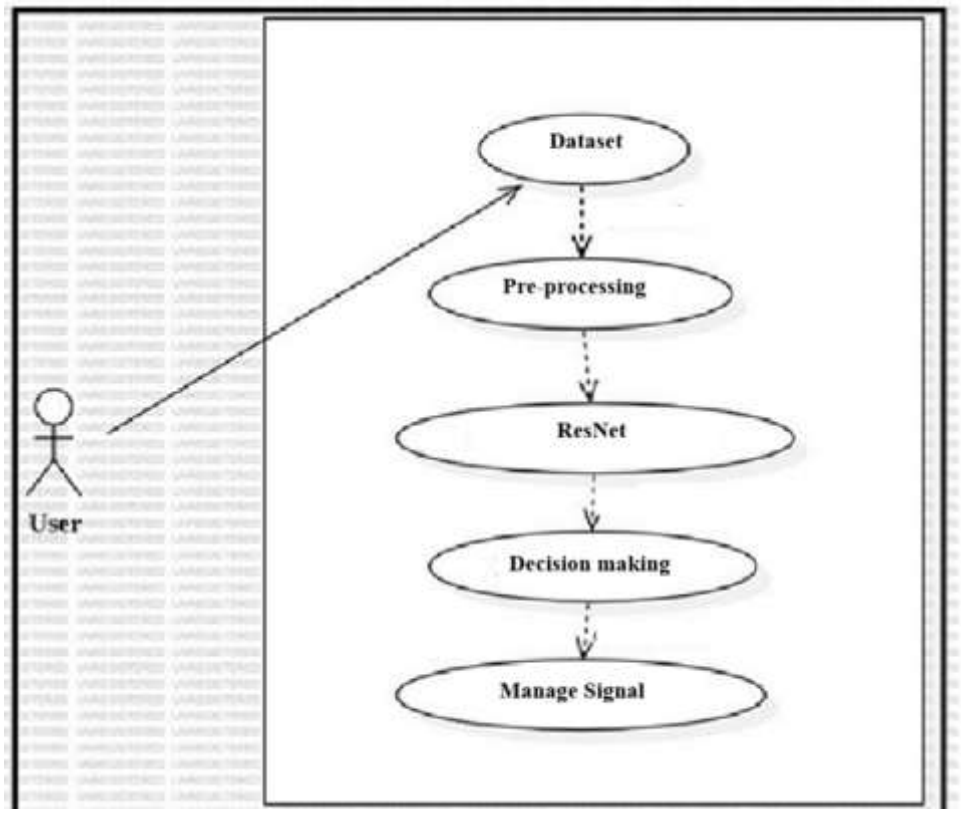


Fig 6 Usecase Diagram

The Use case Diagram depicts the various use cases that are performed by the user in the proposed model. The use cases include dataset, pre-processing, image normalization, decision making, manage signal.

5.1.4 Sequence Diagram

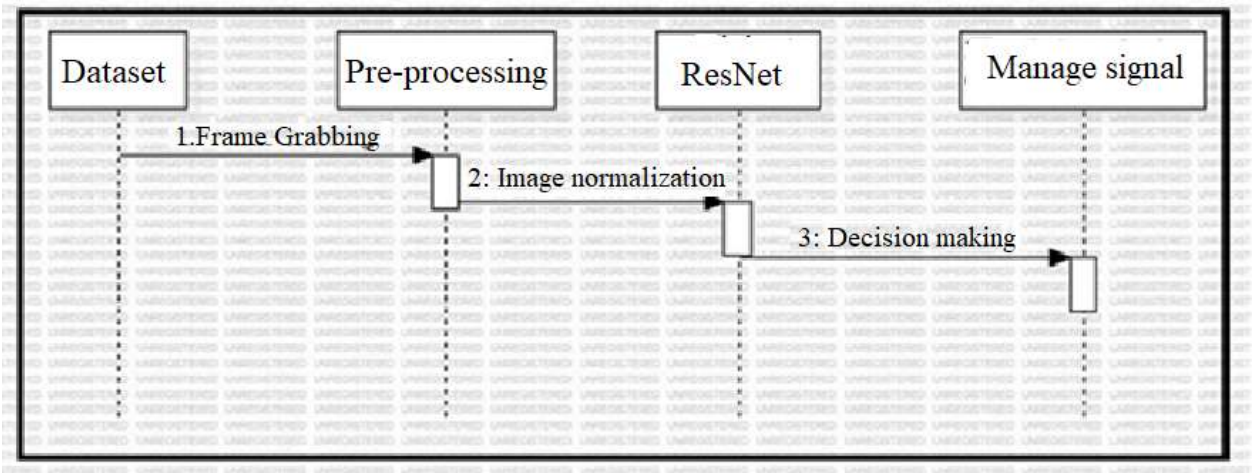
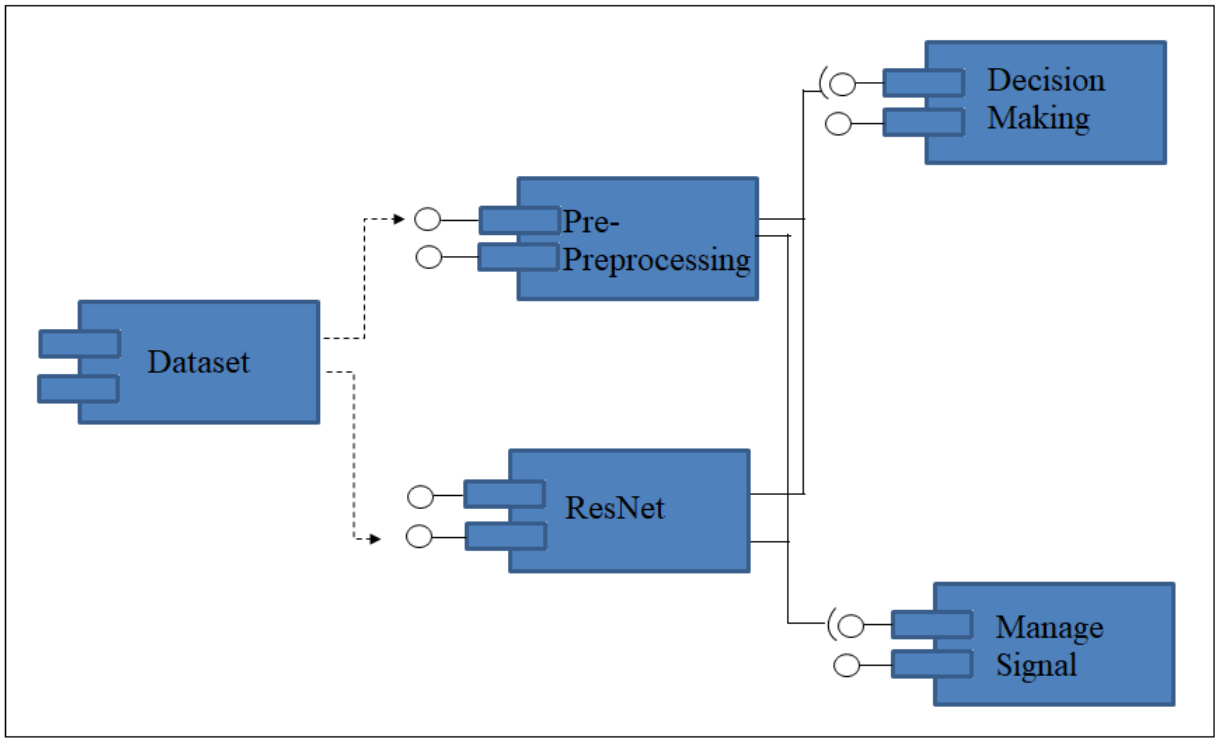


Fig 7 Sequence Diagram

The sequence role diagram provides a sequence of the approaches as well as the various roles performed in the intermediate. In this dataset is provided to frame grabbing, after which pre-processing, is performed and result goes to image normalization then ResNet is applied and perform the decision making to get manage signal.



5.1.5 Component Diagram

Fig 8 Component Diagram

The component diagram illustrates the important components in the proposed system. In our approach the important components consist of the dataset which is interlinked with pre-processing and ResNet, these two modules are further linked to the decision making manage signal.

5.1.6 Deployment Diagram

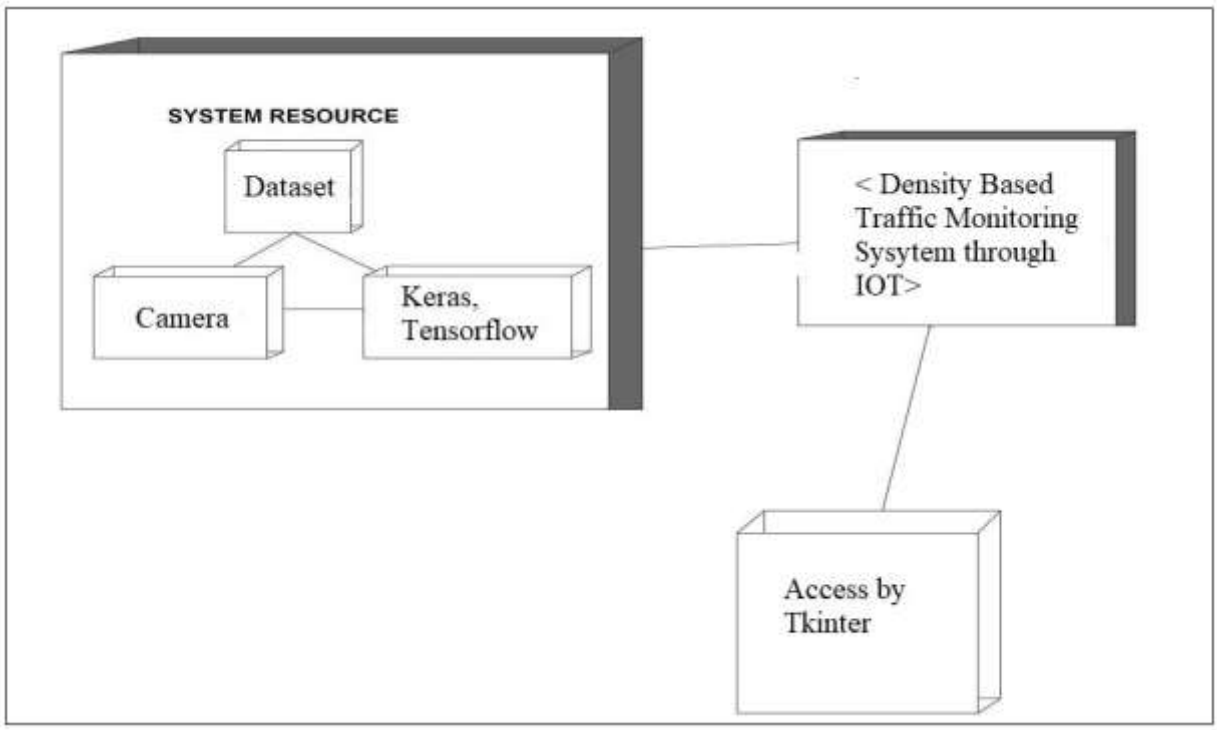
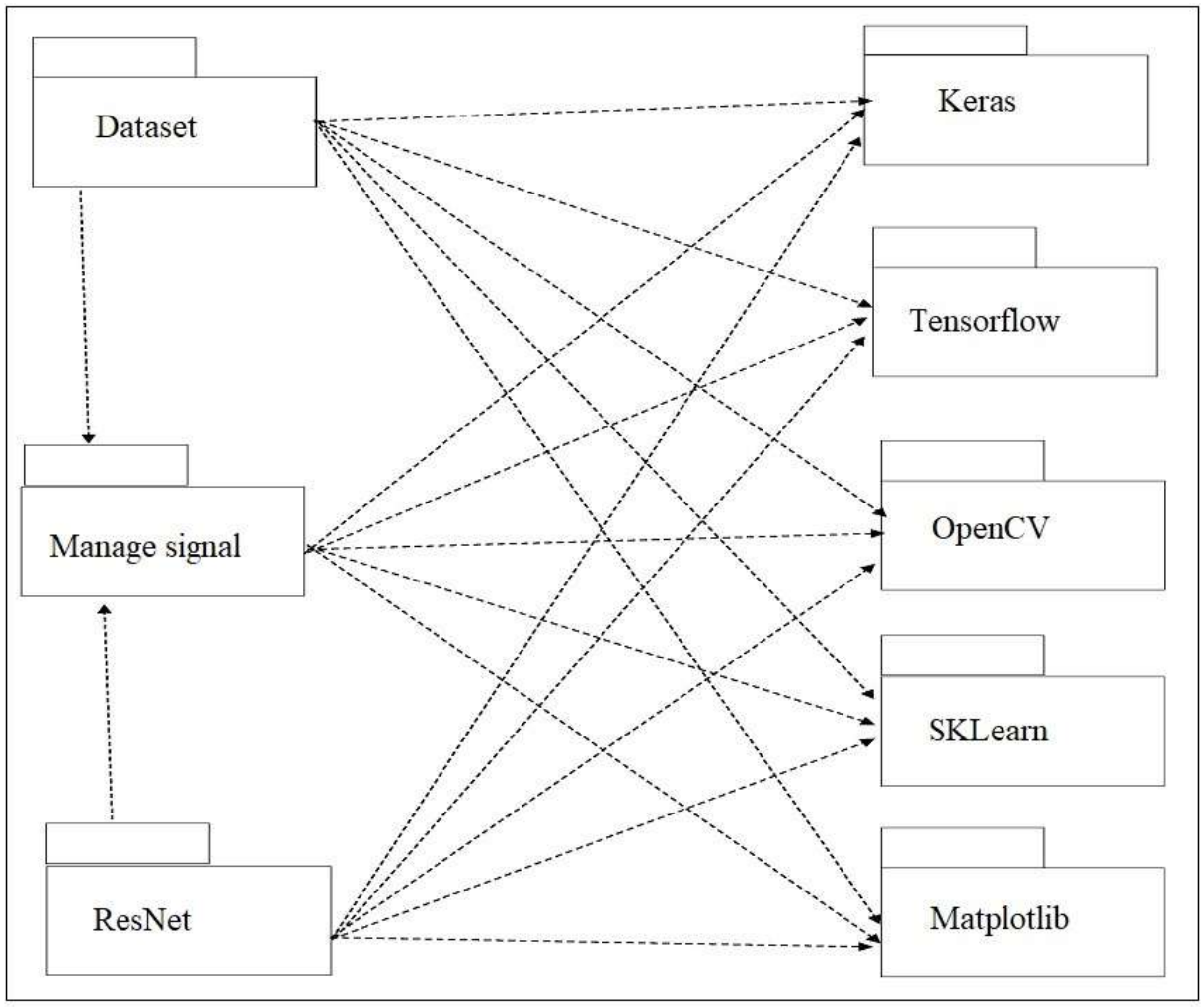


Fig 9 Deployment Diagram

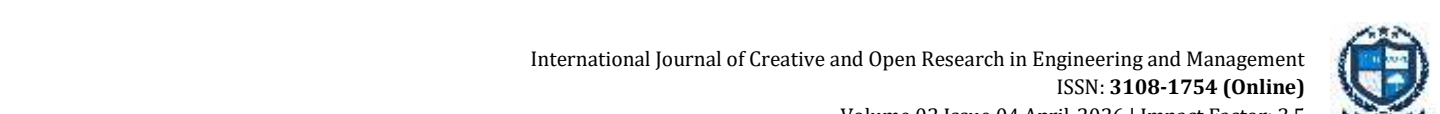


The deployment diagram illustrates the important resources that are utilized for the deployment purposes. In our approach the system resources consist of dataset, camera and the keras, and tensorflow libraries along with the Density based traffic monitoring system through IOT.

5.1.7 Package Diagram

Fig 10 Package Diagram

The package diagram lists the important modules and the relevant packages that are interconnected with each other. The important modules include, Dataset, Manage signal and ResNet and the packages include keras,, tensorflow, opencv, sklearn, matplotlib.



5.1.8 State Transition Diagram

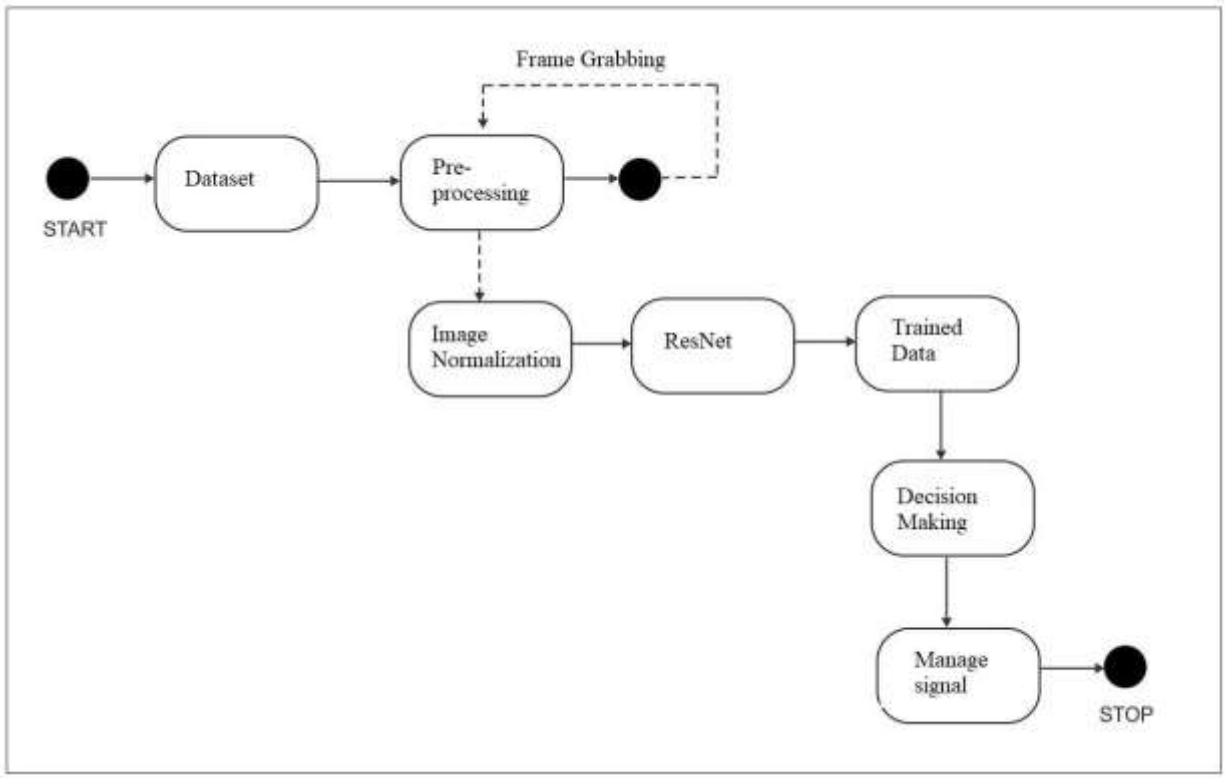


Fig 11: State Transition

The state transition diagram provides the various states that the proposed system goes through. Initially the start state wherein the dataset is provided to preprocessing and then frame grabbing is done. After which preprocessing result is provided to image normalization and image resizing is done, then ResNet is applied to get trained data after which if-then rules is performed which implements the decision making to get manage signal and then reaches the stop state.

5.1.9 Action Plan

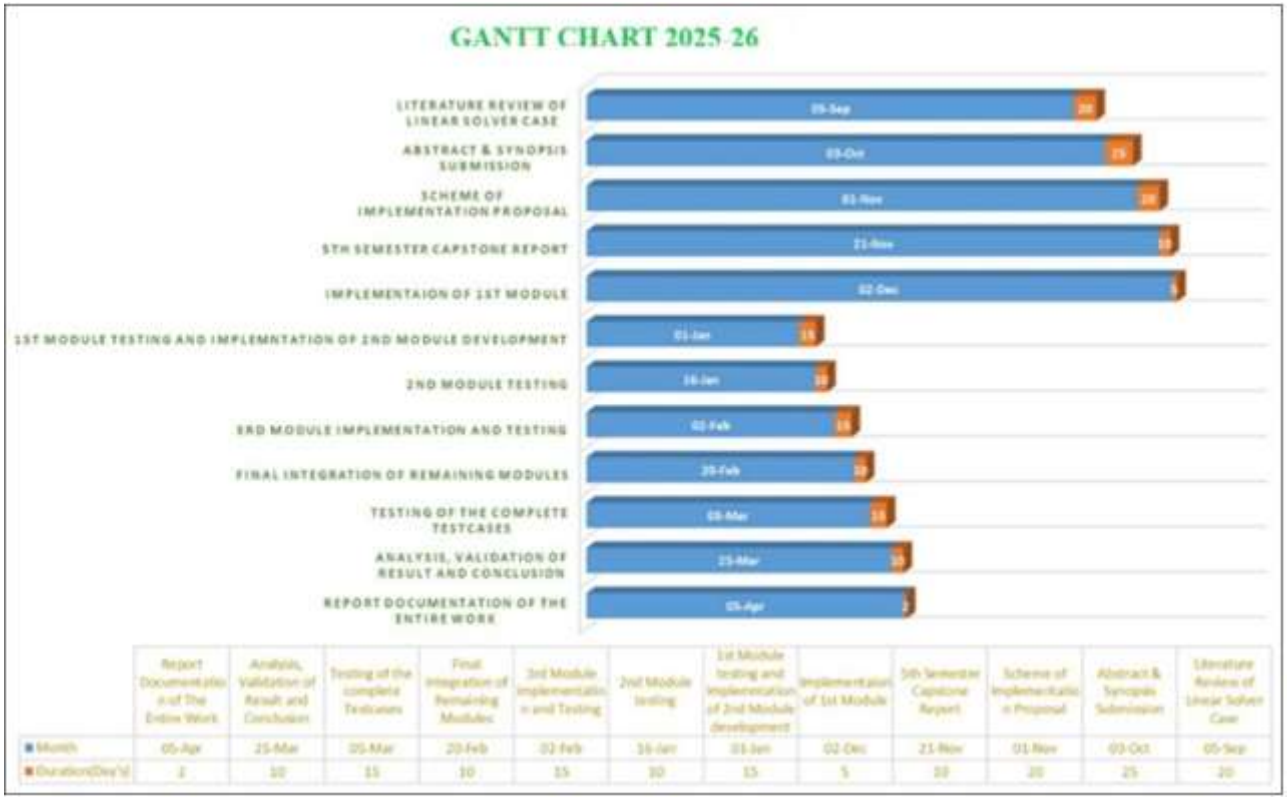


Fig 12: Action Plan

5.2 WORKING AND PROCESSES

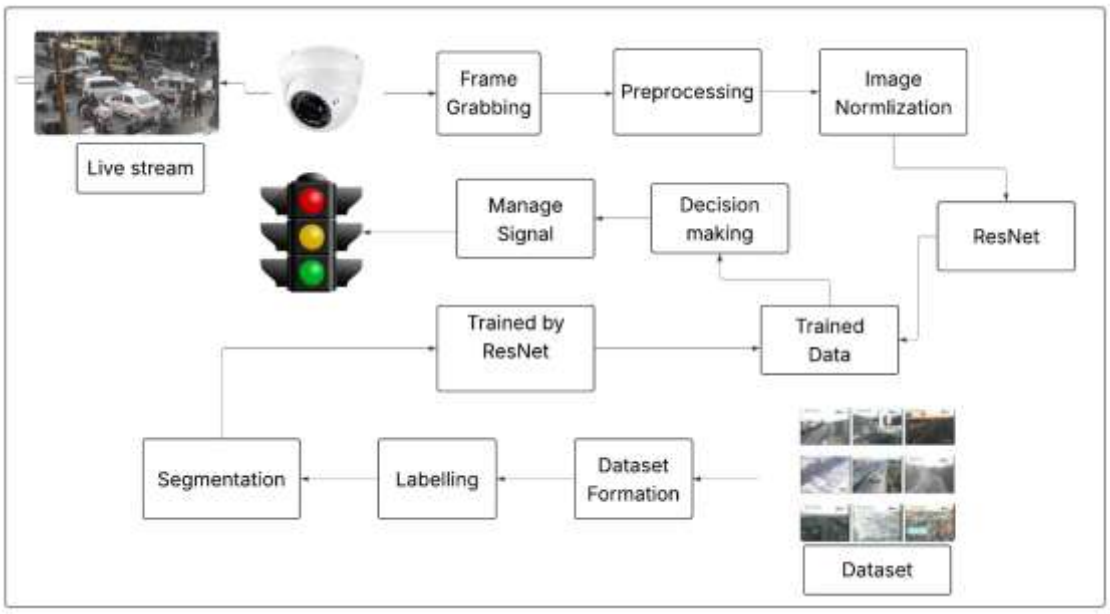


Figure 13 System Overview

The proposed methodology for Density Based Traffic Monitoring System through IOT is depicted in the figure 13. And the steps that involve in this process is broadly explained with the below narrated steps.

Step 1: Dataset Generator: The initial stage involves utilizing OpenCV to capture images of the different vehicles passing by on the road. The Video Capture method in the CV2 package takes photographs of the targeted traffic with the purpose of detecting them during model testing. The dataset folder contains an exhaustive collection of traffic. The model's current emphasis is on working on ambulance-related models. The obtained images are saved to a folder in advance of the model's subsequent step.

Step 2: Data Labelling*:* At this point, the photographs gathered in the previous step are given labels using the labeling program. The user can construct the x1, y1, and x2, y2 of the bottom right corner of the rectangle by importing the image into the labeling software and indicating the coordinates of the top right and bottom left, respectively. Saving the obtained coordinates in an.xml file allows the model to be trained using Resnet, a deep learning network.

Step 3: Installation of APIs: The TensorFlow Object Detection API needs to be installed before this Google Colab instance can be used. Clone the [TensorFlow models repository](https://github.com/tensorflow/models) and follow the installation instructions to get this done. To execute the following code snippets, just hit the play button. Installing the conda environment is the next step after downloading and extracting cuDNN files. Then, clone the tensorflow models repository file that you found on GitHub. Object Detection API installation followed.

Step 4: Upload Image Dataset and Prepare Training Data: Here, we will prepare the TensorFlow training data by importing our training images and executing the scripts to generate TF records. After all of the photos have been uploaded, we may sort them into training, validation, and testing folders. The scripts to generate TF Records from our data can then be executed. First things first, on our local PC, please create a folder called "images.zip" and then bundle all of our training images and XML files into it. Verify that the files can be found within the zip folder. In order to configure our picture directories and extract the zip file, we need to run specific commands after uploading. These folders are created in the /content subdirectory of the file system in this case. The "Files" icon on the screen's left side provides access to the file system.

Step 5: Split images into train, validation, and test folders : Find our "images.zip" file—the folder icon on the left side of the screen—among the listed ones. The further steps following the upload of the dataset are to unzip it and create folders to house the images. These folders are created in the /content subdirectory of the file system in this case. The "Files" icon on the screen's left side provides access to the file system.

Following this, you'll need to sort the images into three categories: train, validation, and test. Here is what each set is meant to accomplish: Follow the path: This same set of images served to train the model. Each training

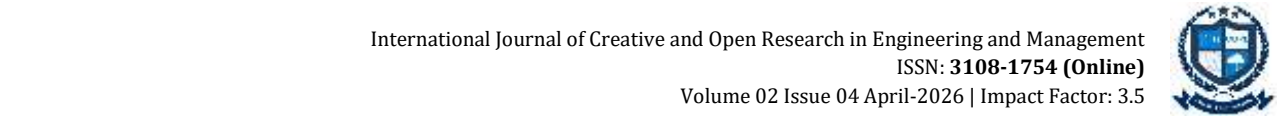
step involves feeding the neural network a batch of images from the "train" collection. After the network classifies the objects in the pictures, it can forecast where those objects are. After the training algorithm calculates the loss, it modifies the network weights using back propagation.

For validation purposes, the training algorithm can make use of the "validation" collection of images to check the progress of training and adjust hyperparameters such learning rate. Unlike the "train" images, these photos are only used occasionally during training, more precisely, once every certain number of steps. Be warned that these images will never be seen by the neural network during training. A human should use these to check the model's accuracy in the end.

Step 6: Create TF Records : Converting the images to TensorFlow's training data file type, TF Records, is the final step. The use of Python programs allows for the automated conversion of data to TF Record format. Prior to running them, we need to create a class label map. Below is the code component that can be used to build a class list in a "labelmap.txt" file. Replace the 'class1','Class2' text with our classes, such as "Ambulance" ,"Fire Engine", and add a new line for each class. Press the play button to execute the code. As a result, the object detection model is told which classes to search for in a "labelmap.txt" file.

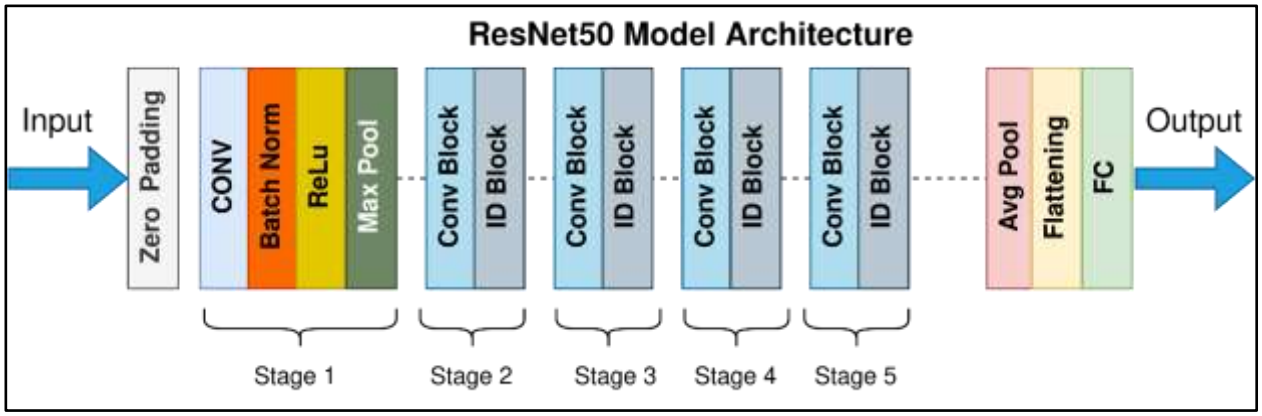
Step 7: Set Up Training Configuration : Here, we set up the processes necessary to train an SSD-Resnet model. The TensorFlow 1 Object Detection Model that will serve as our foundation is named here. Each model comes with its own configuration file that does a lot of things, like telling you where to find specific files and letting you change training parameters like learning rate and total steps. In order to modify the custom training job, we are editing its configuration file. The first section of the code displays the available models in the TF1 Model Zoo. See the names of the files used to download the model and configuration file after that. As more models are added in the future, it will be much easy to keep track of which one is being utilized using this configuration. In the "chosen_model" field, provide the name of the model that we wish to train with. The model named "ssd-resnet-v1-quantized" has been chosen for usage. The next three stages involve downloading the pre-trained model and configuration file after you've specified their parameters.

Adding some generic training parameters to the configuration file is the next step after downloading the model and configuration files. During the training phase, the following factors are controlled: amount of stages: The total amount of steps used to train the model. Forty thousand steps is a reasonable starting point. We can add more stages if we observe that the loss measurements are still decreasing after training is finished. An increase in the number of stages causes training to take more time. Furthermore, if the loss level approaches a plateau, training can be stopped before the specified amount of steps. batchesize: The number of images used in each training run. A larger batch size reduces the number of steps needed to train a model, but it is limited by the amount of GPU RAM that is available for training. For Colab instances, 16 GPUs is the recommended minimum. (Intended solely for quantization-aware training) number of quant_delay_steps Following these extensive procedures, the training algorithm will introduce "fake" quantization nodes into the network in an attempt to mimic quantization. A reasonable starting point would be to cut the overall number of training steps in half. For



further information, see [this article](https://neuralet.com/article/quantization-of-tensorflow-object-detection-api-models/). Here you also supply supplementary training data, like the overall class count, the path to the configuration file, and the file holding the pre-trained model. Changing the configuration file is necessary to implement the newly defined training parameters. The following code will automatically modify the downloaded.config file to incorporate our unique parameters, allowing us to generate our own "pipeline_file.config" file. To make a unique configuration file, add our dataset, model checkpoint, and training parameters to the standard pipeline file. In the block that follows, the training script is updated to save checkpoints every 1000 steps. We can adjust 'num_eval_steps' to save checkpoints more or less frequently as needed. Table 1 displays the resnet architecture.

Figure 8.2: Model Summary for ResNet



Step 8: Train Custom TF1 Object Detector: The training process for our item detection model has started! This model is trained using the "model_main_tf2.py" script that is part of the TF Object Detection API. You have already defined all of the arguments and parameters that'model_main_tf2.py' needs in earlier sections of this Colab. The actual amount of time needed for training can vary between two and six hours, depending on factors such as the model, batch size, and total number of steps. This will create a.tflite file for use in model testing.

Step 9: Testing the model for Traffic Density Monitoring : Here, the Python program makes advantage of the cross-platform Droid Cam app to capture video and, by implication, still images taken by the phone's camera. A file that holds the model that has been trained. We use tflite to find the traffic density and then the lights on the traffic signals turns into green based on the density level.

5.3 Hardware Specification

5.3.1 ESP 32

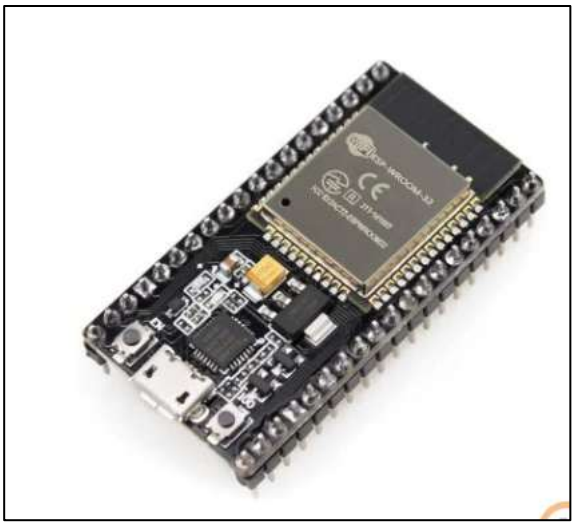


Figure 14 : ESP 32

5.3.2 ESP 32 Specification

Based On:	ESP32 Based, WROOM Type
Flash Memory (MB):	4MB
Processor	Single or Dual core Tensilica Xtensa 32-bit LX6
Operating Voltage (VDC):	2.3 ~ 3.6
Operating Current (mA):	80
Clock Frequency (MHz)	80 to 240
Data Rate (Mbps)	54
Operating Temperature (°C):	-40 °C ~ 85 °C
Dimensions (L x W x H) mm	5 × 2.5 × 1.5mm
Net Weight	9g
Shipping Weight	0.012 kg
Shipping Dimensions	9 × 6 × 2 cm

Figure 15 : ESP 32 Specification

CHAPTER 6

RESULT AND APPLICATIONS

6.1 RESULTS

PLEASE PUT THE SCREENSHOTS OF YOUR PROJECT HERE AND CONTINUE THE FIGURE NUMBERS

(PUT AS MUCH AS CAN)

6.2 Applications

- ✓ SmartCities
- ✓ Highways

6.3 TEST CASES

6.1.1 Performance Testing

The performance of the system is evaluated based on the accuracy of traffic density detection from live video frames using the ResNet model.

6.1.2 System Testing

The system is tested for stability by processing continuous live stream frames and ensuring smooth signal management under high traffic conditions.

6.1.3 Recovery Testing

The system can be restored within a short time after failure by reloading the trained model and reconnecting IoT devices to resume operations.

6.1.4 Security Testing

6.1.4.1 Stress Testing

The system is tested with a large number of video frames to check its ability to handle peak traffic load without performance degradation.

6.1.4.2 Unit Testing

Each module such as frame grabbing, preprocessing, ResNet model, and decision-making is tested independently for correct output.

6.1.4.3 Black Box Testing

The system is validated by providing input video streams and verifying correct traffic signal decisions without analyzing internal code.

6.1.5 Integration Testing

All modules including IoT sensors, image processing, and signal control are integrated and tested together to ensure accurate traffic management.

6.1 Test Cases and Test Results

6.1.2 Test Cases

ID	TEST CASE	INPUT	PASS CRITERIA
TC_01	Frame Processing	Live video stream	Frames are captured and preprocessed without delay
TC_02	Traffic Density Detection	Input frames	System correctly detects vehicle density using ResNet
TC_03	Signal Decision Making	Density data	Traffic signal changes based on detected density
TC_04	System Integration	Full system input	All modules work together and produce accurate output

Table 9.1: Test Cases

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The Density Based Traffic Monitoring System through IoT using ResNet provides an intelligent and automated solution for modern traffic management. By integrating IoT-enabled cameras with the ResNet deep learning model, the system effectively detects and analyzes real-time vehicle density to dynamically control traffic signals. This approach significantly reduces congestion, waiting time, and fuel wastage while improving overall road efficiency and safety. The system represents a major step toward smart city development, offering a scalable and sustainable solution for efficient urban transportation management.

Future Work

Future work includes integrating advanced AI models for more accurate traffic prediction and real-time decision-making. The system can also be enhanced with smart city infrastructure and emergency vehicle prioritization for improved efficiency.

CHAPTER 8

APPENDIX

8.1 Density Based Traffic Monitoring System through IOT

- ✓ https://ieeexplore.ieee.org/document/8359445
- ✓ https://link.springer.com/chapter/10.1007/978-3-030-03131-2_34
- ✓ https://www.sciencedirect.com/science/article/pii/S1877050920309169
- ✓ https://www.irjet.net/archives/V7/i5/IRJET-V7I51234.pdf
- ✓ https://www.researchgate.net/publication/343456789_IoT_Based_Smart_Traffic_Management_Syste

CHAPTER 9

REFERNCES AND BIBLIOGRAPHY

[1] S. Upadhye, S. Neelakandan, K. Thangaraj, D. V. Babu, N. Arulkumar and K. Qureshi, "Modeling of Real Time Traffic Flow Monitoring System Using Deep Learning and Unmanned Aerial Vehicles," in Journal of Mobile Multimedia, vol. 19, no. 2, pp. 477-496, March 2023, doi: 10.13052/jmm1550-4646.1926.

[2] S. Rafiq and M. A. Khanum (2021) Review on Minimization of Ambulance Response Time Using Image Processing and Critical path Mapping Based on Traffic Control, Vol. 02, Iss. 02, S. No. 002, pp. 1 7, 2021. https://doi.org/10.54060/JIEEE/002.02.002

[3] Patel, R., Mange, S., Mulik, S. *et al.* AI based emergency vehicle priority system. *CCF Trans. Pervasive Comp. Interact.* **4**, 285–297 (2022). https://doi.org/10.1007/s42486-022-00093-7

[4] KHERRAKI, Amine; EL OUAZZANI, Rajae. Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring. IAES International Journal of Artificial Intelligence (IJ-AI)**,** [S.l.], v. 11, n. 1, p. 110-120, mar. 2022. ISSN 2252-8938. Available at: <https://ijai.iaescore.com/index.php/IJAI/article/view/21104>, doi:http://doi.org/10.11591/ijai.v11.i1.pp110-120.

[5] Sunil M, V Yashaswini Naidu, Vignesh R, Vishwas P, Amitha S, " SMART TRAFFIC MANAGEMENT FOR AMBULANCE, " Volume:04/Issue:12/December-2022 Impact Factor- 6.752 www.irjmets.com

[6] Usaid, M., Muhammad Asif, Tabarka Rajab, Munaf Rashid, & Syeda Iqra Hassan. (2022). Ambulance Siren Detection using Artificial Intelligence in Urban Scenarios. *Sir Syed University Research Journal of Engineering & Technology*, *12*(1), 92–97. Retrieved from https://sirsyeduniversity.edu.pk/ssurj/rj/index.php/ssurj/article/view/467

[7] Bhoomika G M et al., "Ambulance Detection using Image Processing ," DOI 10.48175/IJARSCT-5667

[8] Mahmud, Umar, Hussain, Shariq, Sarwar, Amber, Toure, Ibrahima Kalil, A Distributed Emergency Vehicle Transit System Using Artificial Intelligence of Things (DEVeTS-AIoT), *Wireless Communications and Mobile Computing*, 2022, 9654858, 12 pages, 2022. https://doi.org/10.1155/2022/9654858

[9] Yarra Kavitha, Penke Satyanarayana, Shafi Shahsavar Mirza, Sensor based traffic signal pre-emption for emergency vehicles using efficient short-range communication network, Measurement: Sensors, Volume 28, 2023, 100830, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2023.100830.

[10] Dr. P. Sankar Babu, K. Meenendranath Reddy, P. Naga Timmaiah, & V. Srikanth. (2023). Intelligents Traffic Light Controller for Ambulance. *Journal of Image Processing and Intelligent Remote Sensing*, *3*(04), 19–26. https://doi.org/10.55529/jipirs.34.19.26

[11] Rawand Sulayman, Salih Rajab and Bawer Kareem. Emergency Vehicle Detection with Computer Vision. *ScienceOpen Preprints.* 2023. DOI: 10.14293/PR2199.000508.v1

[12] Intelligent Traffic Signal Prioritization for Emergency Vehicle Diversion in Urban Environments using Multi-modal Deep Learning - Santosh D. Pandure, Pravin L. Yannawar - IJFMR Volume 6, Issue 4, July-August 2024. DOI 10.36948/ijfmr.2024.v06i04.23981

[13] Amrutasagar, K.; Manoj, Pera; Divya, Morla; Mahesh Babu, Meethukulla; Gangotri, Lavudiya, **"**Enhanced Traffic Signal Adaptation with Ambulance Identification and Distance Computation**,"** International Journal of Computing and Digital Systems ISSN (2210-142X) Int. J. Com. Dig. Sys. #, No.1, 1-10 (March-2024)

[14] Noor, A.; Algrafi, Z.; Alharbi, B.; Noor, T.H.; Alsaeedi, A.; Alluhaibi, R.; Alwateer, M. A Cloud-Based Ambulance Detection System Using YOLOv8 for Minimizing Ambulance Response Time. Appl. Sci. 2024, 14, 2555.

****