ABSTRACT

The project aims to solve the problem of traffic congestion at junction signals by dynamically scheduling signal timings using machine learning and real time data processing. This enhances traffic efficiency, reduces congestion, and improves overall transportation system performance.

We are introducing DTSA (Dynamic Traffic Scheduling Algorithm). The dataset would involve taking an image of a junction from google maps with the congestion levels being represented by different colours. We proceed by taking this image of the map surrounding the signal, then logically slicing the circle into 3 sectors, based on pixel count. This logical slicing of map is going to be converted into a 2d array form to be given to an SVM model for scheduling. LSTM is also going to be added to this model for pre-emption of signals. We will be using Simulation of Urban Mobility (SUMO) as our simulation tool to help us analyse the traffic flow after we have provided it with optimised signal timings that are obtained by our machine learning model.

CHAPTER I

I. INTRODUCTION

In today's rapidly growing world, the problem of traffic congestion has just been deteriorating day by day. Traffic congestion has emerged as an enormous challenge to overcome as it poses significant damage to society. Not only does it increase the amount of harmful toxins and contribute to pollution and global warming but also pose a serious threat to urban mobility and economic productivity of a country. Road rage and accidents are another side effect of this problem. With the population increasing at a fast rate and cities expanding, the strains on transportation infrastructure have become more pronounced. This brings us to a dire need of innovative solutions to resolve this problem and enhance the transportation management system.

Currently the traffic management system follows the traditional approach of managing traffic signal timings by having predetermined times for all signals without considering the real time changes in the congestion levels of a lane. Traffic patterns keep changing on a regular basis and require an evolved solution to manage it in order to ensure smooth vehicle movement. By following the static approach there is no promotion of sustainable urban development. This problem requires advanced technologies and new strategies to enable optimisation of traffic signal operations.

Our project addresses this issue of traffic congestion through the innovation of an intelligent and adaptive traffic signal scheduling system. Making use of machine learning algorithms and real time data processing, we aim to change the way traffic signals are controlled. This approach makes them dynamically adjust the timings depending on the congestion levels in the lanes of a four-way junction. This would make the traffic flow smoother and efficient and help in overall development of the traffic management system.

We passionately aspire to be able to make a difference by committing to innovation and sustainability. By using emerging technologies, we endeavour to create a traffic management system that helps improve the situation currently and overcomes the challenge we face on a day-to-day basis. Through collaborative efforts we aim to redefine the future of mobility in urban and suburban areas and offer a forward-thinking solution to this complex challenge.

CHAPTER II

II. PROBLEM STATEMENT

Chronic traffic congestion levels have become a serious challenge which we need to overcome. It only leads to increased travel times, fuel consumption and environmental pollution. It hampers the growth of a country. The traditional system which is currently in use consists of either manually moving traffic or giving predetermined signal timings for all lanes irrespective of how much the congestion level is. The challenges of this system are particularly evident during peak hours when the congestion levels keep changing frequently. It lacks the smartness to respond quickly and effectively. Moreover, controlling the traffic flow manually is labour-intensive and hard to apply.

This has become outdated now and we need to ensure we use the correct technologies and make use of the advanced methodologies we have at hand now to upgrade this system and make it efficient and smart.

The primary objective we have is to create a smart and adaptive signal control system which dynamically changes the timings depending on the real time congestion levels in the lanes of a junction. This is made possible by using real-time traffic data processing and applying certain machine learning models. We follow an approach that makes sure our system is capable of improving the overall traffic management system and promotes environmental sustainability.

We plan to address the issues highlighted above by not relying on the use of highly sophisticated sensors and only use images. These images will be our input data and will be taken from Google Maps. The images will highlight the congestion levels at the lanes of a junction using different colours. These images are then processed and the congestion level information is converted into a 2D matrix which is then given to our machine learning models. These models generate the optimised signal timings for the traffic

signals for all the lanes at the junction. Simulations using SUMO are carried out for us to observe and monitor the performance before deploying it in real life signals.

This proposed system will not have a high maintenance cost and does not require any specialised training and staff to use it after it has been trained and deployed. The proposed approach will revolutionise the traditional traffic control system and create a more efficient and adaptive signal control system.

CHAPTER III

III. LITERATURE SURVEY

3.1 Prediction of Switching Times of Traffic Actuated Signal Controls Using Support Vector Machines

3.1.1 Objective of this paper

This paper aims to develop an algorithm for the prediction of traffic actuated signal controls and present its mathematical foundations.

3.1.2 Implementation and Methodology

- Multiple Cycles: Results in 86% accuracy, which beats older methods.
- Super Short-Term: Accuracy shoots up to 97%, but this needs super-fast, high-quality data.
- Conclusions
- SVMs Work: They seem well-suited for predicting traffic light changes.
- Data Matters: How they structure the traffic data makes a huge difference in how accurate the predictions are.
- Driver-Friendly Predictions: For this to be useful in a driver assistance system, they need to turn predictions into a "chance of green" readout.

3.1.3 Achievements of the paper

- Predicting several light changes ahead based on overall traffic volume
- Predicting just the next change using updated traffic volume
- Super short-term prediction within the current light cycle based on individual cars being detected

3.1.4 Limitations of the Paper

- Traffic Isn't Predictable: Some lights follow simple schedules; others change completely based on how busy the roads are. Naturally, predicting the unpredictable lights is harder.
- Data Delays: It takes time to collect traffic data and make it usable. This delay makes it difficult to 'see' the traffic situation in real-time, which is important for accurate predictions.
- The accuracy for one of the proposed methods is over 95% which indicates that it is overfitting.

3.1.5 Validation Methods used

They test this on real-world data from a German intersection.

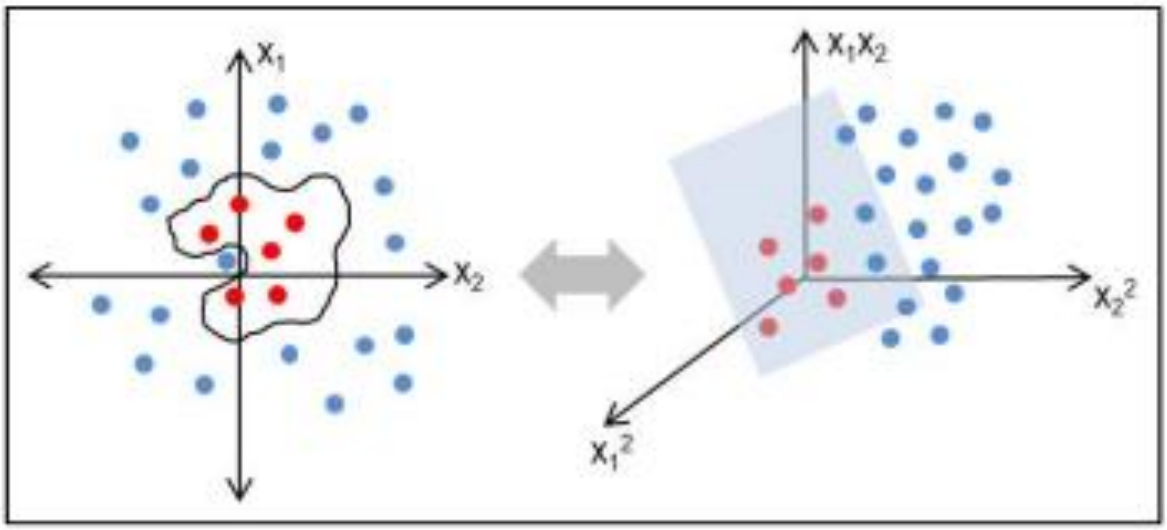

Figure 3.1 - Representation of a hyperplane

3.2 Deep Q Learning with LSTM for Traffic Light Control

3.2.1 Objective of this paper

This study investigates the potential advantages and benefits of combining short-term memory (LSTM) recurrent neural networks and deep-cue networks (DQN) for real-time, dynamic traffic light control (TLC) The assumption is that DQN understands LSTM of traffic patterns over time Decision-making will be improved, resulting in more efficient traffic flow compared to a DQN-only TLC system.

3.2.2 Implementation and Methodology

- Traffic modelling platform: SUMO (Simulation of Urban Mobility) is used to model common intersections with configurable traffic characteristics.
- Comparative Benchmark - The proposed hybrid TLC system (DRQN-TLC) is compared with a conventional DQN-based TLC controller without LSTM.
- Metrics: Traffic flow efficiency is evaluated using the following metrics.
- Cumulative awards (reflects the reduction in traffic at the intersection);
- Average travel time of vehicles
- Vehicle waiting time throughout the intersection
- Neural Network Architecture

3.2.3 Achievements of the paper

- The AI system combines MLP (Multi-Layer Perceptron) and LSTM using Keras and TensorFlow algorithms.
- It takes advantage of the DQN algorithm, for learning matching experience replay and objective networks.

3.2.4 Input and Output

- Input: Simulated ‘tracking’ from a virtual camera monitoring the intersection, providing data on vehicle presence, speed and current traffic light phase
- Output: ‘Actions’ limit

3.2.5 Validation Methods used

The study uses a simplified simulation scenario, which may not capture the full complexity of a real car crash, including unpredictable driver behaviour, sensor failure, and the street is there an assumed chaos of traffic detection camera-based accurate vehicle detection

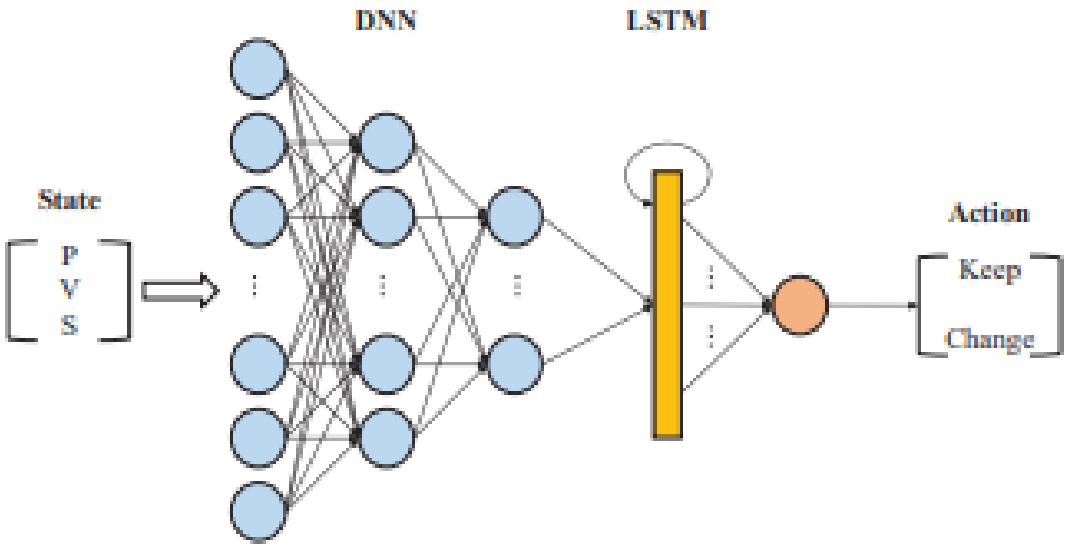

Figure 3.2 - Structure of DRQN-TLC

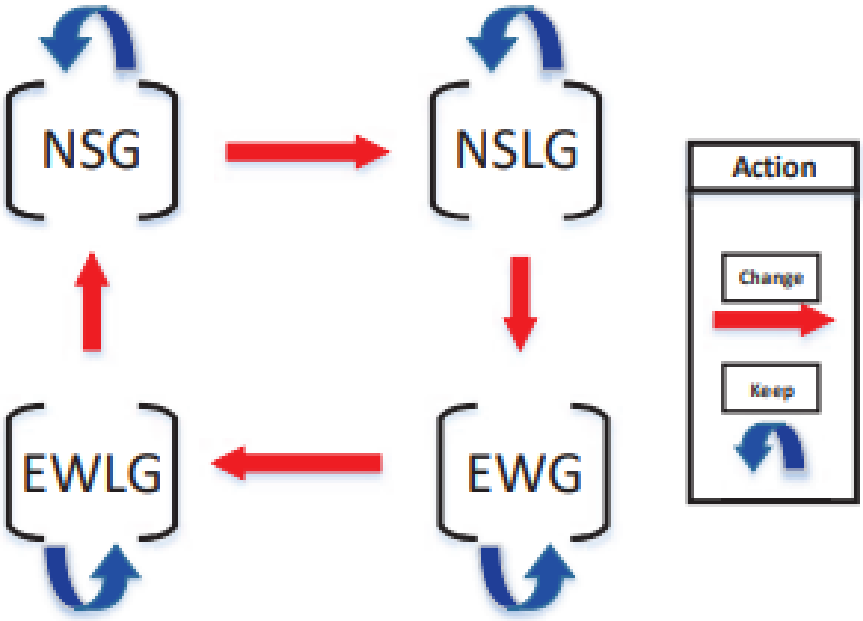

Figure 3.3 - Sequence of traffic light changes

3.3 Smart Traffic Detection and Control using Machine Learning Techniques

3.3.1 Objective of this paper

This paper uses vehicle detection methods through surveillance cameras and machine learning to predict traffic based on historical data in a region. Finally, optimizing traffic by utilizing the predicted data and proposed methodology reduces the average wait time of travellers.

3.3.2 Implementation and Methodology

- Detection: Employ YOLO object detection algorithm to identify and count vehicles (cars, buses, etc.) in intervals of 5 minutes.
- Prediction: Historical traffic data is pre-processed and used to train an MLP model for each lane.
- Optimization:
- Predicted vehicle volume is calculated for each lane.
- Green signal time is allocated proportionally to the predicted traffic, with busier lanes receiving longer green times.

3.3.3 Advantages of the paper

- Dynamically allocates signal time for a given lane using a traditional algorithm.
- A Machine Learning (ML) model, specifically a Multi-Layer Perceptron (MLP), predicts traffic volume based on historical data.

3.3.4 Limitations of the Paper

- Done only for one lane at a time, i.e., only 1 lane is free at a time and not multiple, the way it works in the real world
- Done only for a up-front POV

3.3.5 Validation Methods used

- A custom traffic simulator was developed to compare the proposed system against fixed-time signalling.
- Performance Metric: Average vehicle delay (i.e., time spent waiting at the intersection).

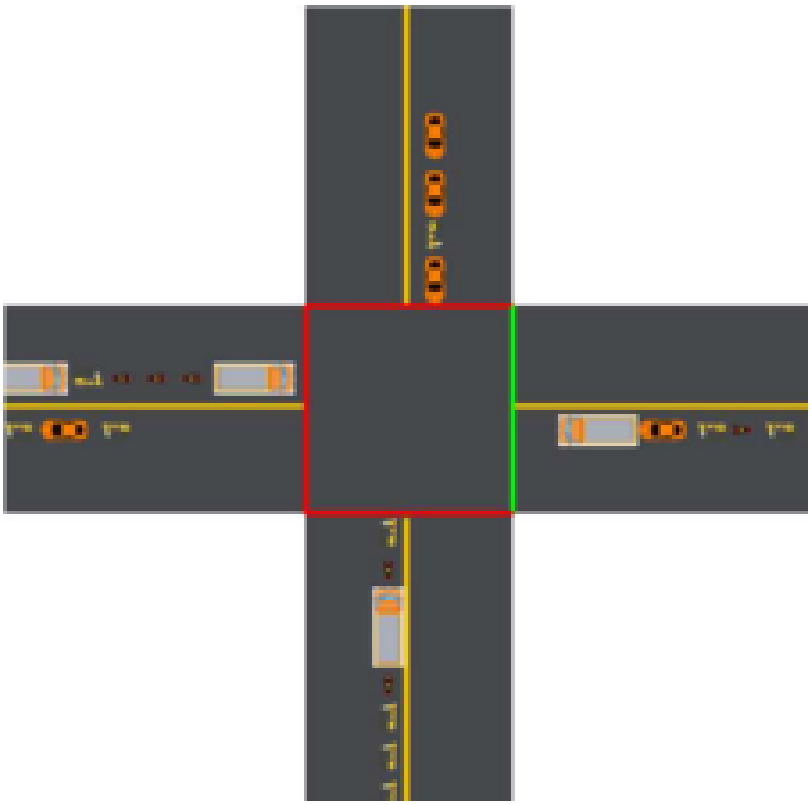

Figure 3.4 - Simulation design

3.4 A Hybrid Model for Music Genre Classification Using LSTM And SVM

3.4.1 Objective of this paper

To Build a music sorting system that can figure out the genre of a song. Specifically, combining two types of AI models to make this system more accurate than using each one individually.

3.4.2 Implementation and Methodology

- The LSTM was trained to find patterns in how the music changes over time.
- The SVM was trained to make strong genre classifications.
- To classify a new song, add up the "confidence scores" from both modes and go with the maximum value

3.4.3 Achievements of the paper

- The combined system nailed the genre of songs 89% of the time, which was better than either model could do by itself.
- Possible Speed Boost: While not measured directly, the SVM part probably makes the whole process faster.

3.4.4 Limitations of the Paper

- They only take a look at a general overview of the song, not focusing on how it changes dynamically.
- Traditional machine learning can get bogged down with complex music data, either not being accurate enough or slowing to a crawl.

3.4.5 Validation Methods used

- Compared their found accuracy with multiple other models for the same dataset.
- Made a confusion matrix for the same

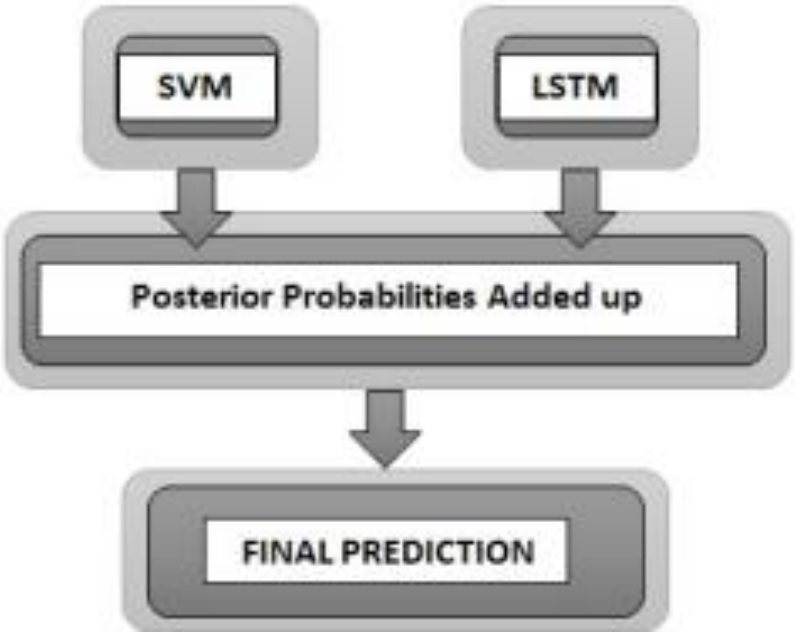

Figure 3.5 - Proposed combined model (LSTM and SVM)

3.5 An accurate traffic classification model based on support vector machines

3.3.1 Objective of this paper

Figuring out traffic flow in a network is tough. We want a smarter way to label that traffic accurately and in real-time – something way better than the old, unreliable methods. Enter our idea: a new approach using Support Vector Machines (SVMs) to nail this classification problem.

3.5.2 Implementation and Methodology

- Scale it Down: Data overload slows everything. They streamline things first.
- PCA: Feature Detective: Not all data is created equal. PCA pinpoints what data actually matters, avoiding needless complexity.
- Particle Swarm Optimizer (PSO): Finding the perfect SVM settings is difficult. The improved PSO automates this, making the SVM shine.
- Multi-Class Mayhem: No more simple binary classifications. They handle the complex, multi-category traffic of the real world.

3.5.3 Achievements of the paper

- SPP-SVM is 98.6% accurate for simpler classifications, and still over 92% even when things get complex.
- Reduced the data needed by almost 90%
- Real-Time Ready: This classifies traffic in under a second.
- Sample Smart: Works great even with small training sets.

3.5.4 Advantages of the Paper

- Untrustworthy Port Number: Old systems relied on port numbers to guess what an application was. The lines between port usage are way too blurry now. This is no longer the case with this paper.
- Deep Packet Inspection: Results in slowdown. Although digging through every packet is useful, it adds massive time delays.
- Standard Protocol Checks: These are Encrypted! Anything outside the norm, especially encryption, throws these methods off.
- Old-Generation AI: Complex and Costly. Traditional machine learning is difficult – tons of data needed, slow, and there is a constant need for changing parameters.

3.5.5 Validation Methods used

- Prepared the Data: Normalized it, split it into training and test sets.
- Key Features Only: PCA, then extra refinement to find the most important data.
- PSO Power: Found those perfect SVM settings no human wants to mess with.
- Training Time: Fed it all the good stuff, out pops a powerful SPP-SVM.
- Test Drive: The trained model gets to work classifying, and we see just how well it does.

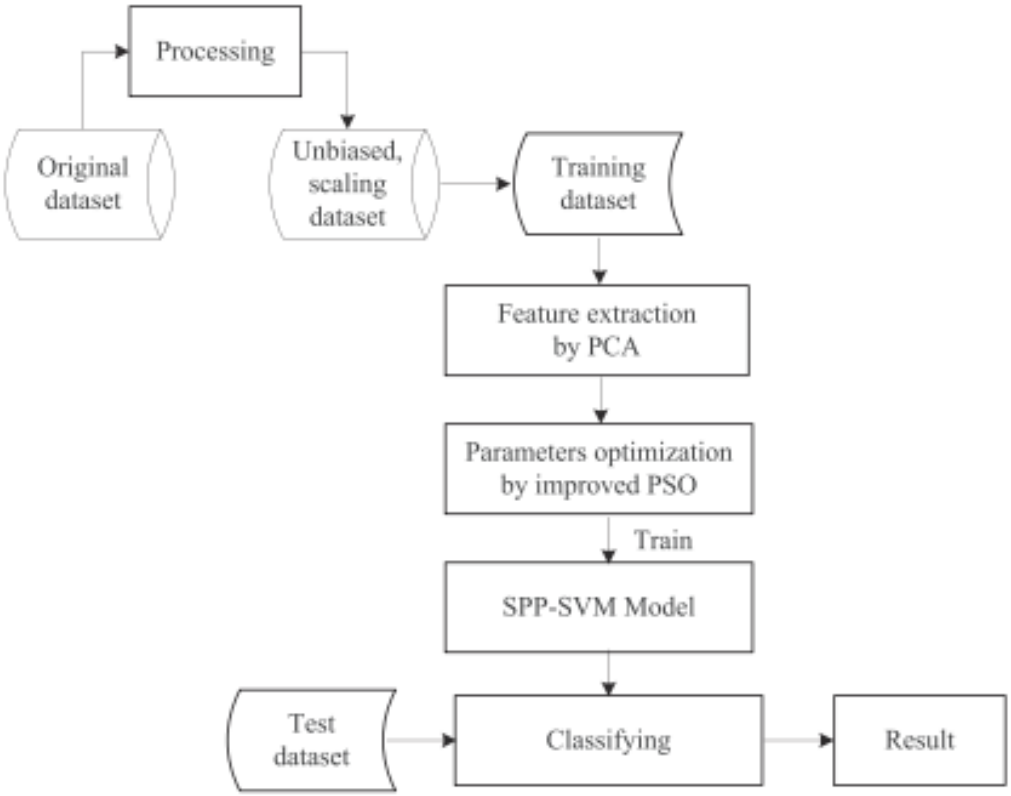

Figure 3.6 - Flowchart of SPP-SVM model

3.6 Image Processing and IoT Based Dynamic Traffic Management System

3.6.1 Objective of this paper

This paper presents an adaptive traffic management system using Internet of Things (IoT) and Image processing. This system reduces the average waiting time and increases the efficiency of traffic clearance. The system also reduces the pollution due CO2 emission and useful in emergency situations, thus being adaptive traffic management using Internet of Things (IoT)

3.6.2 Implementation and Methodology

- The proposed system has capability to analyse real time data using image processing. Using cameras, different lanes are monitored constantly.
- The data obtained from different lanes are examined.
- Detection and counting of the number of vehicles in each lane is done by using image processing.
- The count from each lane is sent to the central processing unit.
- According to the count of vehicles algorithm calculates waiting time for each lane, then the signal lights will be decided.

3.6.3 Achievements of the paper

- Detects and counts number of vehicles from camera
- Calculates wait and go time based on that
- Shorter wait times to improve traffic flow.

3.6.4 Results of the Paper

The findings indicate that the adaptive system significantly reduced average waiting time compared to the traditional approach, especially during periods of high traffic.

3.6.5 Validation Methods used

They built a simulation to compare their adaptive system against traditional fixed-timing signals. The primary performance metric is the average waiting time at the intersection.

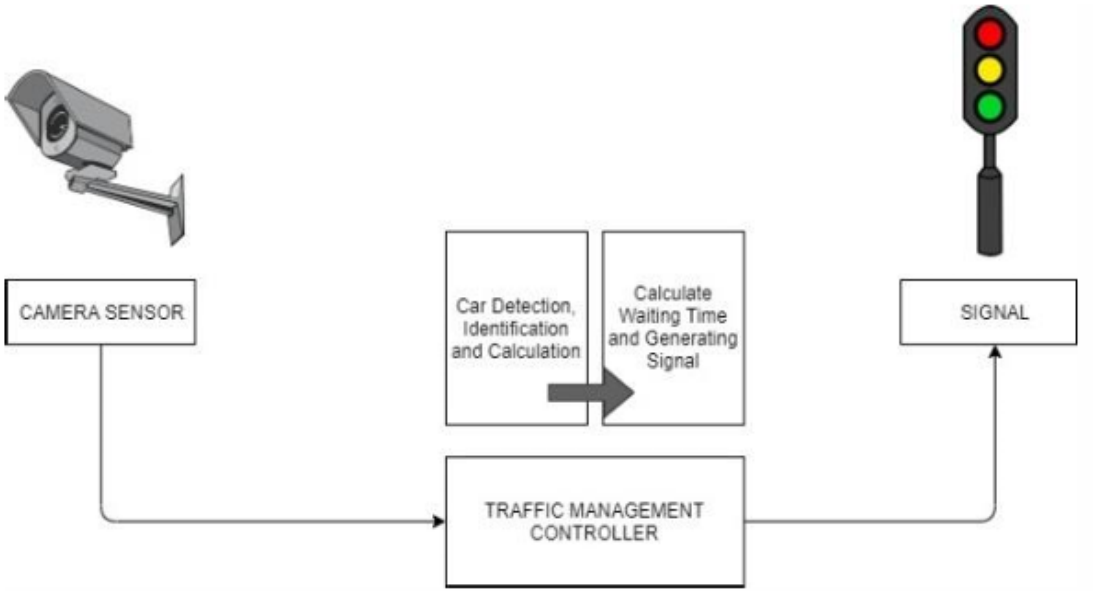

Figure 3.7 - Image Processing and IoT Based Dynamic Traffic Management System Architecture Diagram

CHAPTER IV

IV. DATA

We'll be using Google Maps Data using the Google Maps API in the finished Project.

For Model Training, Testing and Demonstration purposes we'll Primarily use Traffic Image Data from Google Maps along with Traffic Density Matrices that are close to the Traffic Density Matrices we receive from the Google Maps Image Processing.

This is done for 2 reasons:

- To Increase the amount of data we could use to train the model by a significant amount (We expect we'll need over 10,000 Datapoints to train the LSTM)
- To Train the Model with Data that is more varied and would be more useful to generalise it for any traffic signal

Sample Google Maps Image:

Figure 4.1 - Sample Image Datapoint to be used

CHAPTER V

V. PROJECT REQUIREMENT SPECIFICATION

5.1 Product Perspective

Currently there isn't any solution in use which manages the traffic signal timings depending on the real time congestion levels. Our project harnesses the new-age technologies to achieve an improved traffic system

Context and Origin:

- Increase in Urbanisation: leads to increased demands on transportation infrastructure.
- Tech advancements: Our solution aims to provide a responsive traffic control system making use of advancements in machine learning.
- Environmental focus: 'Traffix' will contribute to improve environmental sustainability by minimising congestion which also lowers the pollution caused by the emission of air pollutants from vehicles

5.1.1 Product Features

- Real time Traffic monitoring
- Take an image from google maps which indicates the congestion levels by showing different colours.
- It detects the level of congestion and classifies it as high, medium and low.
- Depending on the congestion our machine learning model provides us with the optimised signal timings

- Traffic simulation software (SUMO) integrated with our project to help us analyse and make our machine learning model more efficient

Overall aim of the project is to reduce congestion levels, minimising delays and enhancing the overall efficiency

5.1.2 User Classes and Characteristics

- Traffic Management Authorities: Access to real time monitoring of the traffic flow given our optimised signal timings
- City planners: This can also be useful to them since they can plan ahead, keeping in mind the traffic congestion levels that is usually present in a specific area
- Regular Commuters: managing traffic signals dynamically is directly beneficial for the regular commuters. It will reduce delay times and likelihood of road rage, giving the regular commuters a seamless experience.

5.1.3 Operating Environment

- The system is designed to operate on server infrastructure, making sure the handling of real-time data processing and adaptive traffic control is done efficiently.
- Compatibility with cloud service instances for ensuring flexible deployment.
- Operating systems can be Windows/Mac/Linux.
- The database used here would be MySql database.

5.2 Assumptions, General Constraints, Dependencies and Risks

5.2.1 Assumptions

- Dynamic traffic scheduling system will not completely replace but complement the working of the traditional traffic control system in its early implementation phase.
- Current system can accommodate this new approach and support communication with the centralised control system
- Real time data from google Maps will be reliable
- Traffic patterns remain relatively consistent in order to obtain useful information while analysing

5.2.2 General Constraints

- The system should follow the traffic rules and guidelines.
- The system must function with the available hardware resources.
- There should be network connectivity in order for our project to work as required.
- Should be compatible with various software integrations.
- Google maps might also show extremely high traffic incase of parked cars with people inside or people walking or standing nearby

5.2.3 Dependencies

- Dependent on getting real time reliable traffic data from google maps.
- Integration with external software such as SUMO for simulation and analysis purposes.
- Strong network infrastructure for data transmission.
- Community able to accept the new approach.

5.2.4 Risks

- Wrong decisions made by our machine learning models if the input data is inaccurate or delayed.
- Resistance by the community may hinder the effectiveness of the result.
- Potential breaches of altering traffic data could cause serious problems and hinder system security.

5.3 Requirements

5.3.1 Functional Requirements

- The input data is in the form of an image taken from google maps.
- Each image has a junction represented on it with the lanes showing the congestion levels using different colours.
- This traffic image data is converted into a traffic data matrix.
- It gives the optimised signal timings using this matrix.
- The system also keeps checking if the traffic signals are online using heartbeats.
- This optimised signal timings is also given to a simulation tool - SUMO which has the junction simulated in it.
- We can observe and analyse the traffic flow to make sure our system is efficient.

5.3.2External Interface Requirements

User Interfaces:

A simple interface for the demonstration of the project. Final project won't have a defined GUI for the users

Hardware Requirements:

- CPU: A processor with capabilities equivalent to or surpassing the Intel Core i5 6th Generation.
- RAM: Minimum 8GB RAM to ensure smooth operation
- Operating Systems: Compatibility with Windows 10 or higher versions for optimal performance.
- GPU: NVIDIA GeForce GTX 960 or a superior graphics processing unit

Software Requirements:

- MySQL Database / Cloud Bucket Storage
- Python 3.10+
- PyTorch OR TensorFlow APIs
- Cloud Compute Architecture

Communication Interfaces:

- Communication standard required is 802.11 Wi-Fi protocol
- We also require a UDP/TCP connection

5.3.3 Non-Functional Requirements

Performance Requirement:

- Capable of real time data processing
- Reliable signal time output

Safety Requirements:

- Don’t let 2 lanes that could probably collide go green at the same time

Security Requirements:

- Should be invulnerable to attacks trying to feed wrong data to models to mess up traffic signal timings
- Dataset should be from accurate source

Other Requirements:

- Portable - Should work on Lightweight Thin Hosts
- Scalability - Architecture should be able to handle multiple similar requests from different traffic signals
- Response Time - Should be capable of real time data processing

CHAPTER VI

VI. SYSTEM DESIGN

6.1 Diagrams

6.1.1 Swimlane Activity Diagram

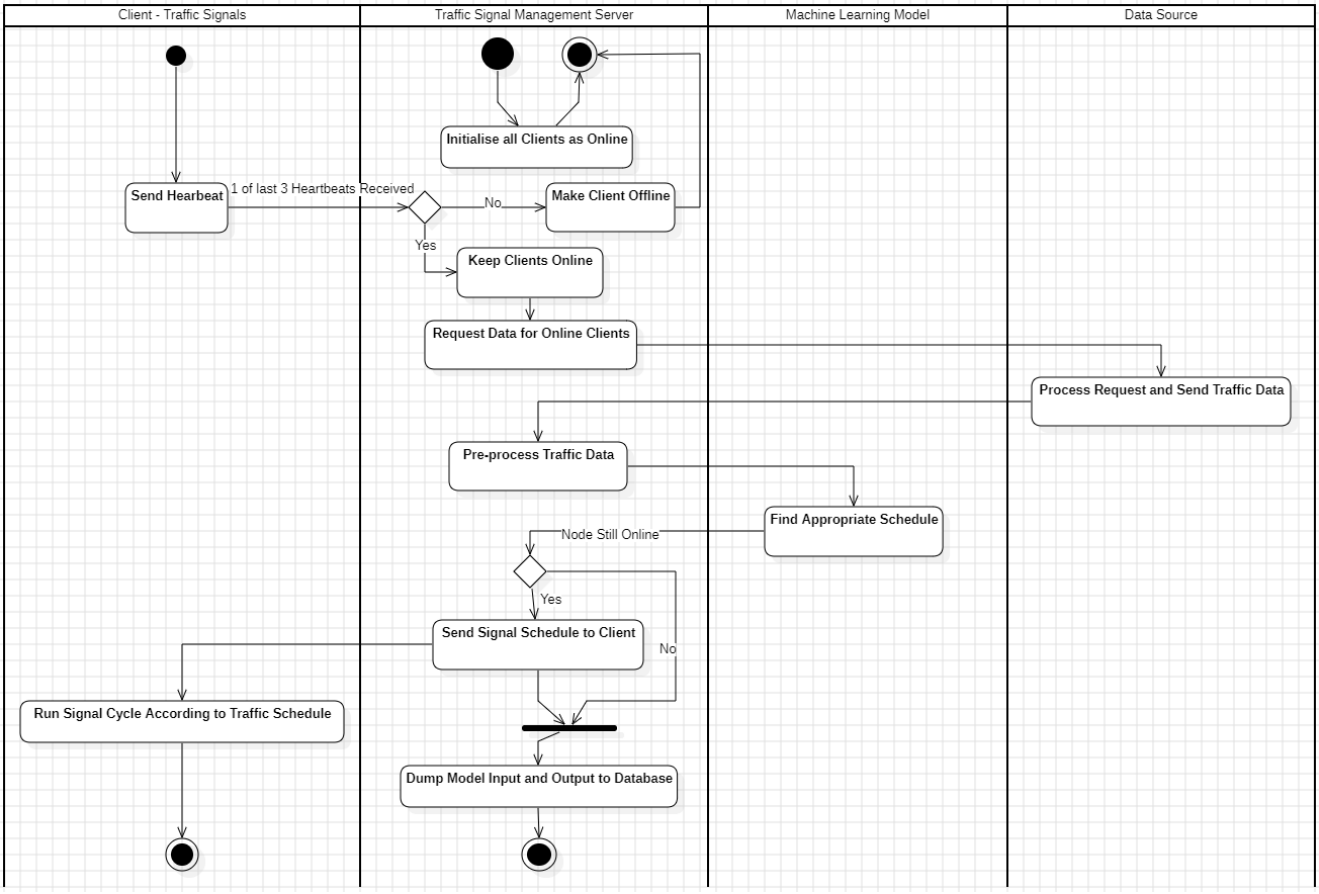

Figure 6.1 - Activity Swimlane Diagram

6.1.2 State Diagrams

Server State Diagram:

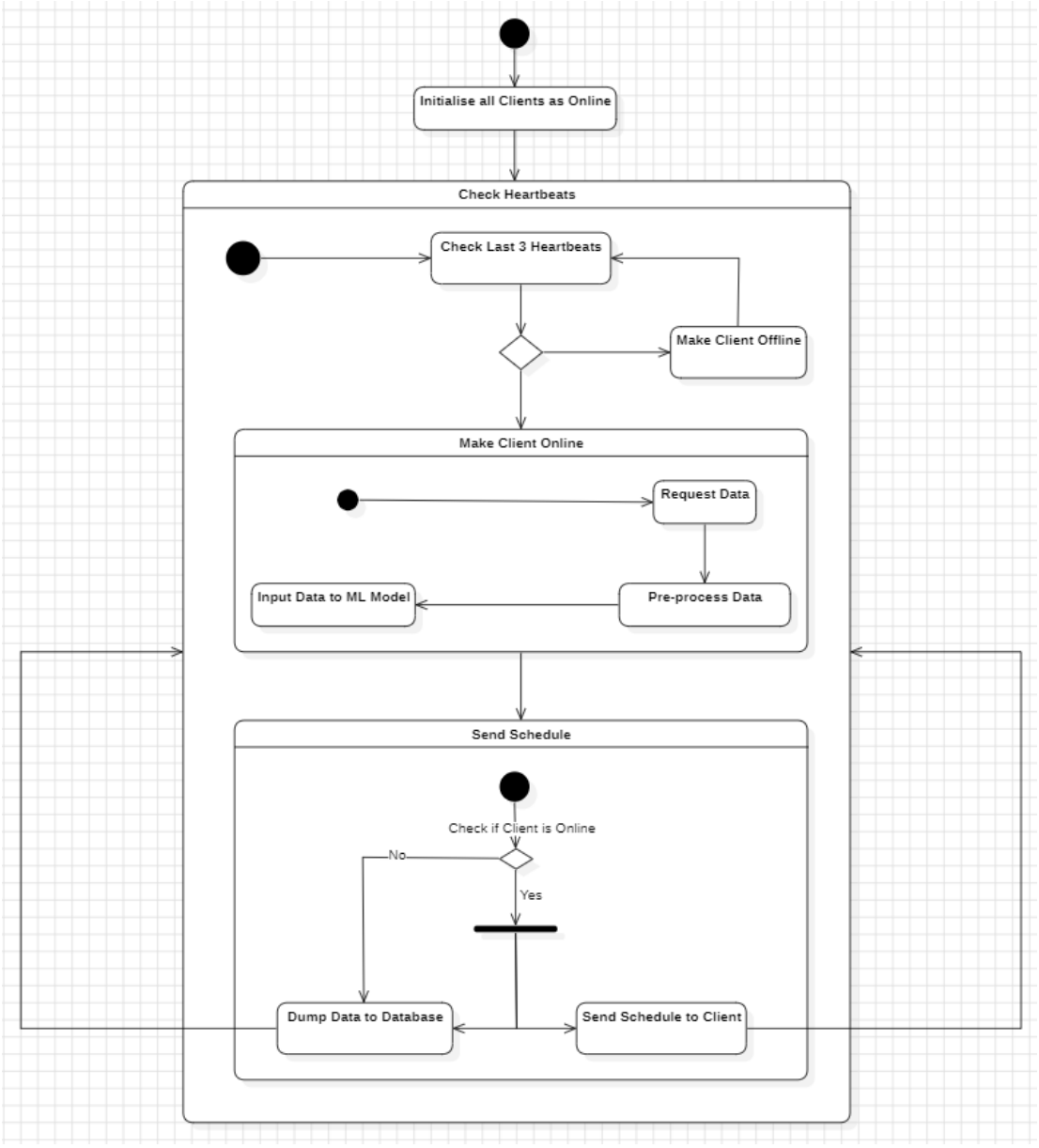

Figure 6.2 - Server State Diagram

Client (Traffic Signal) State Diagram:

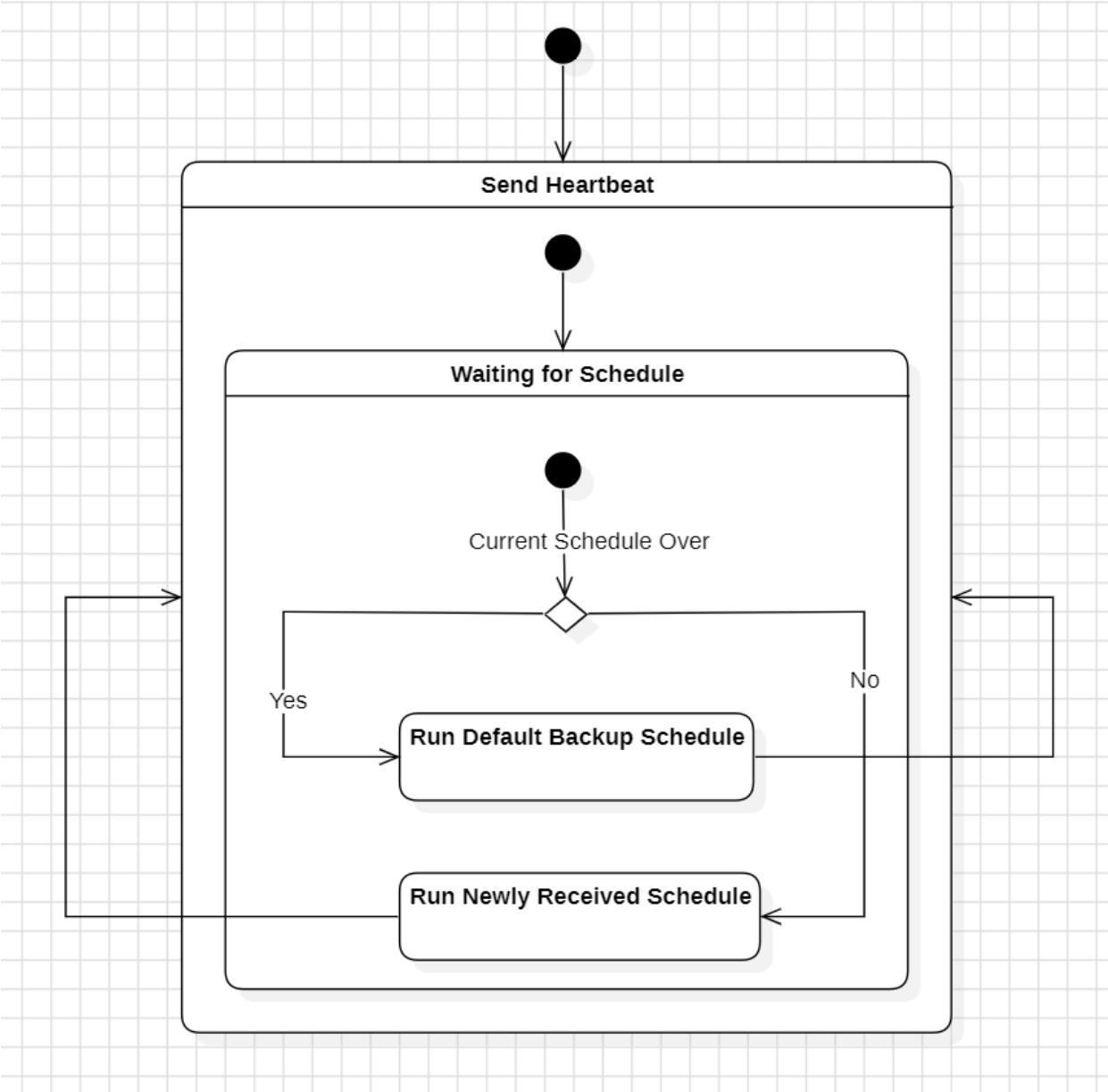

Figure 6.3 - Client (Traffic Signal) State Diagram

6.1.3 Sequence Diagram

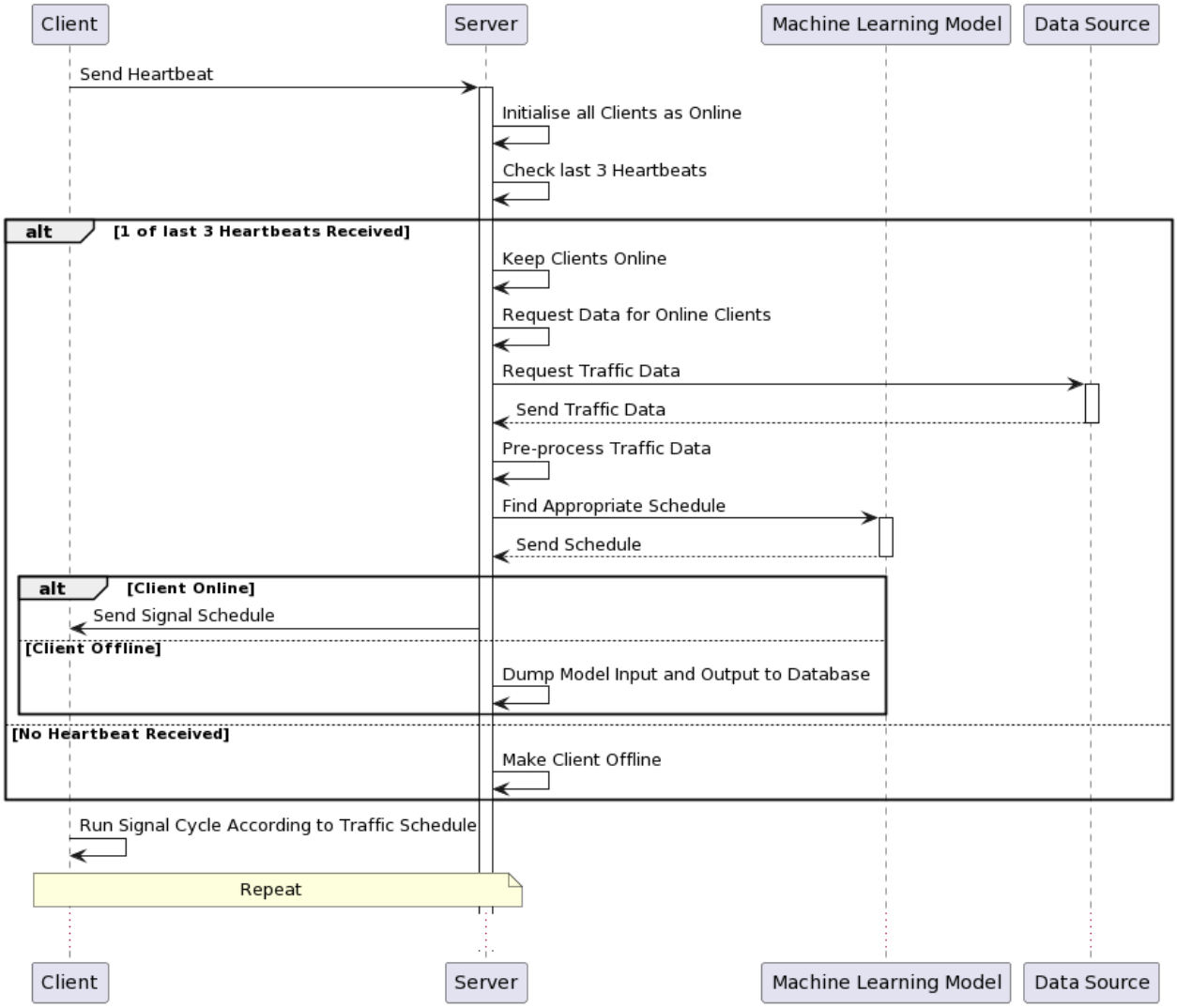

Figure 6.4 - Sequence Diagram

6.1.4 Data Flow Diagrams

Level 0 Data Flow Diagram

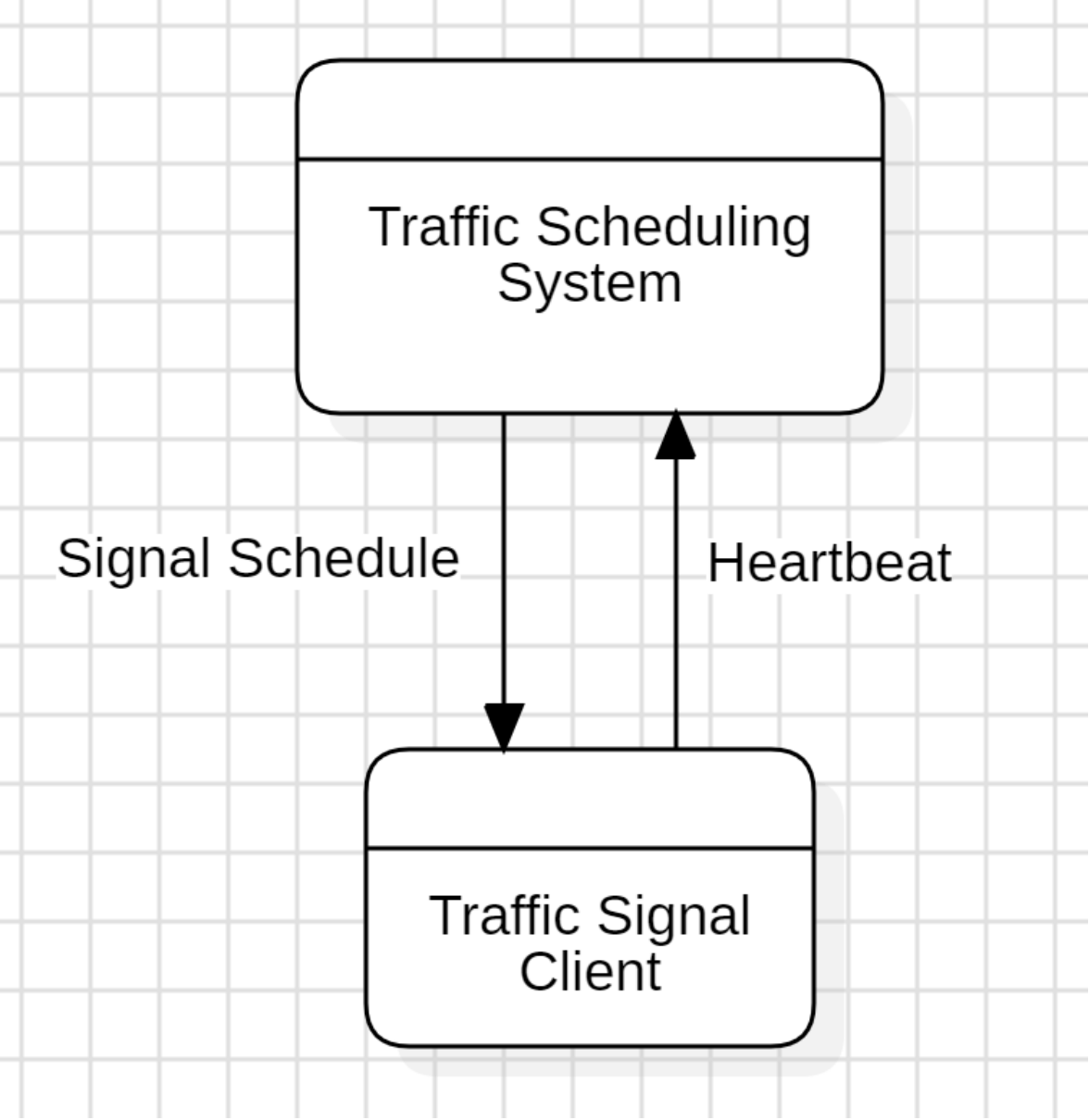

Figure 6.5 - Level 0 DFD

Level 1 Data Flow Diagram

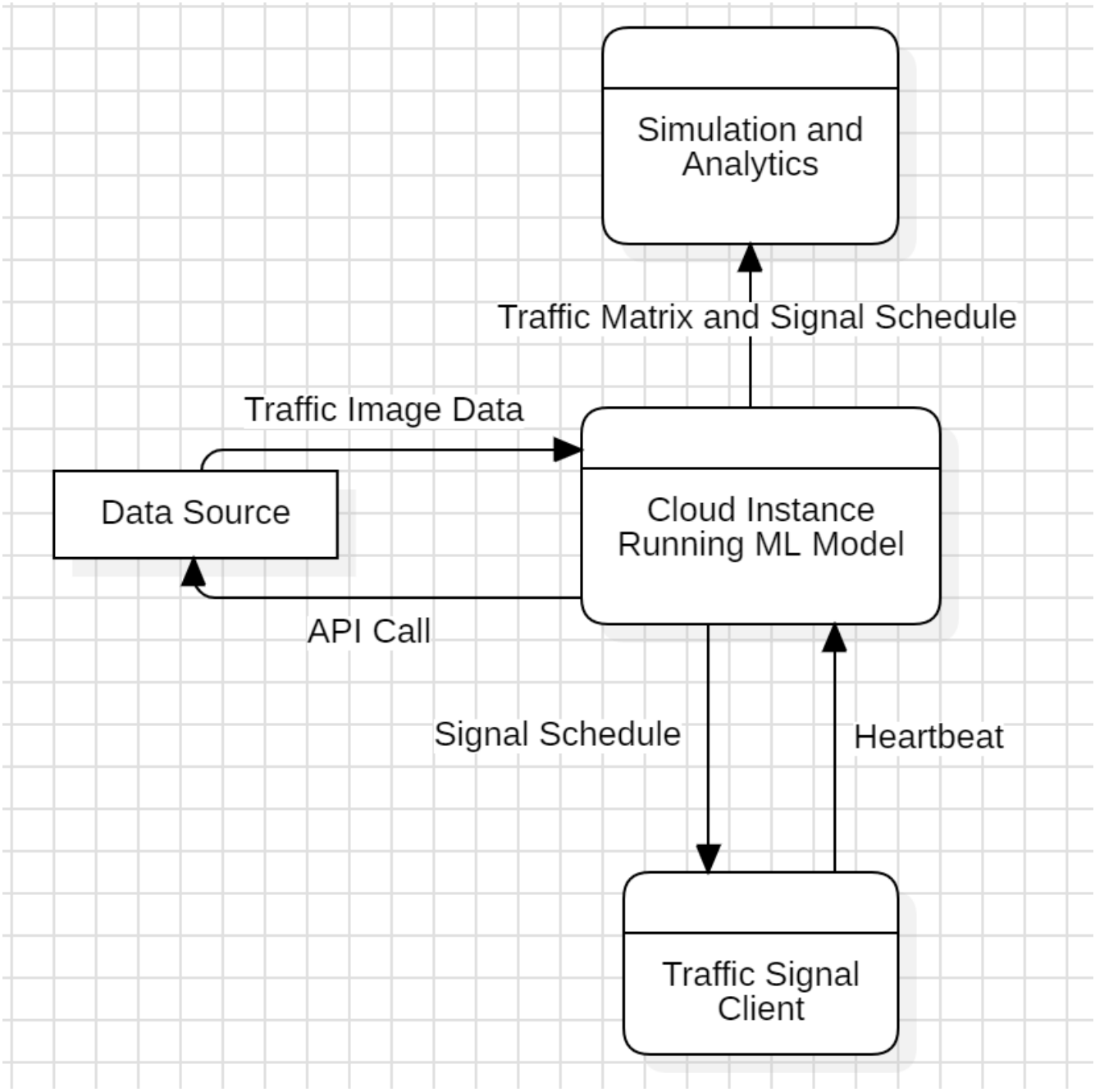

Figure 6.6 - Level 1 DFD

Level 2 Data Flow Diagram

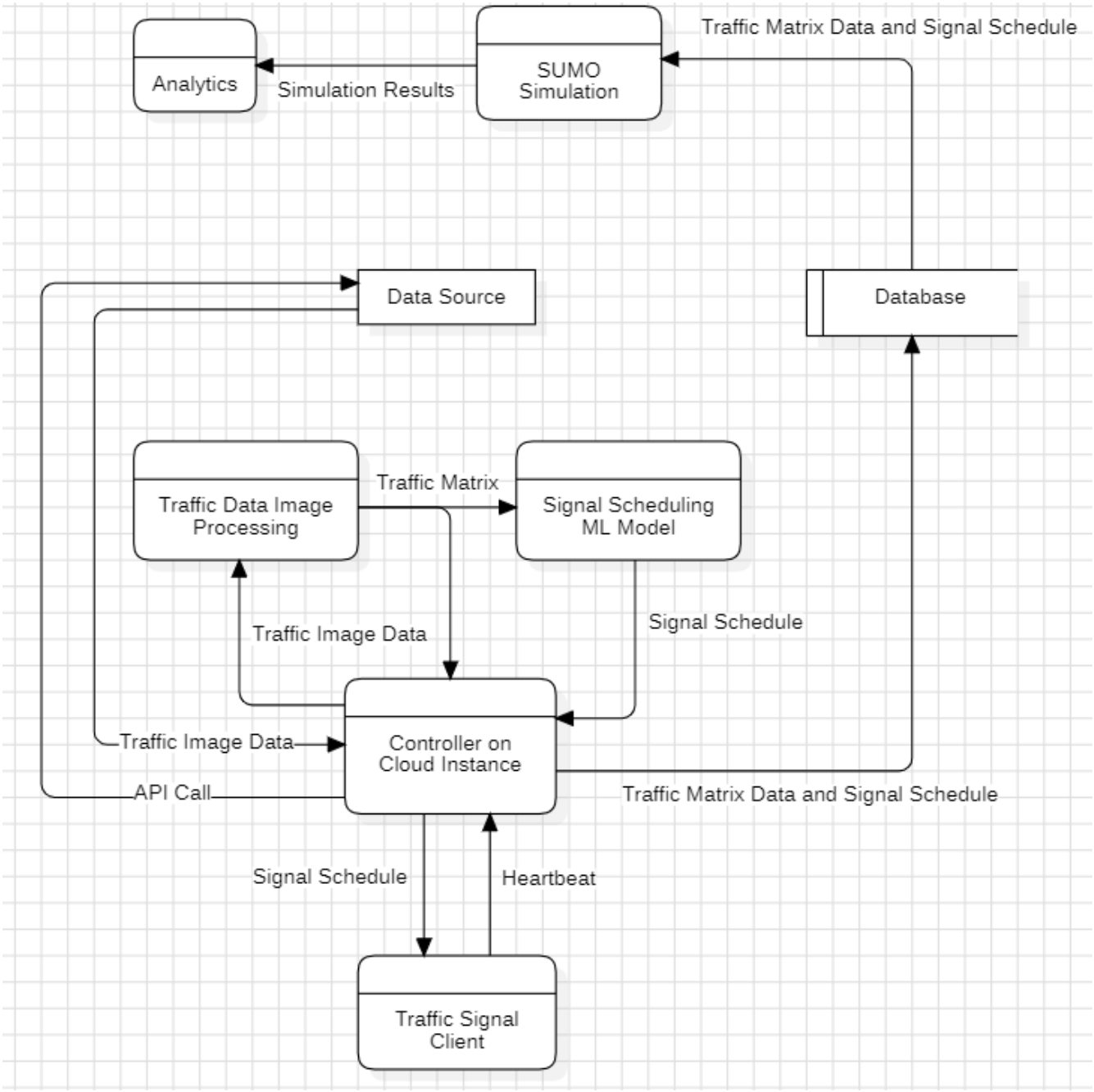

Figure 6.7 - Level 2 DFD

6.2 Architecture Overview

6.2.1 Major Components

- **Traffic Signal Client** – The logic at the traffic signals
- **Controller on Cloud Instance –** The coordinator that takes care of all data transfer and signalling the start of traffic signal algorithm on the Cloud Instance
- **Data Source –** We'll be using the Google Maps API to get Traffic Image Data
- **Traffic Data Image Processing –** Takes input as Traffic Image Data and turns it into Traffic Density Matrix using OpenCV
- **Signal Scheduling ML Model –** Takes input as Traffic Matrix and gives out a Traffic Signal Schedule using SVM and LSTM
- **Database –** Stores Traffic Density Matrix as well as the output Signal Schedule to be Simulated and Analysed Later
- **Simulation –** Uses TraCI and SUMO to Simulate Traffic at the given junction using data from the database
- **Analytics –** Uses Simulation Output to Compare between results of our ML Solution and traditional traffic scheduling using Python's NumPy, Pandas, etc.

6.2.2 Data Flow

The Traffic Signal Client sends a heartbeat to the controller on the cloud instance at specific intervals of time, if this heartbeat isn't received for a certain amount of time, the signal is declared offline on the system.

When a Signal is online, an API call is made to fetch traffic maps data at that junction, which is then fed into the image processing logic and then the ML Model to get the signal schedule. This schedule is then given back to the client to be executed for the next signal cycle

The Traffic Density Matrix along with the Signal Schedule are Put into a Database for further Simulation and Analysis using SUMO and Python

6.3 Traffic Signal Clients

In our system, we decided to make the Traffic Signal Clients lightweight. This design decision was made because it won't be economically viable to have heavy clients (which will run the ML Model on site) at the signal itself.

The client at the traffic signal has the following jobs:

- Send Heartbeats to the Server at equal intervals of time
- Switch Schedule timing to new timing when it's received from the server (Even if it's in the middle of the current signal cycle)
- Switch to Default Signal Schedule when there's no response for server and the signal cycle is over

The Clients will be designed to be deployed on low powered systems that aren't as expensive. This is because these systems won't be doing any heavy processing on site.

6.4 Controller on Cloud Instance

6.4.1 Components of the Cloud Instance

On the Cloud Instance sits the entire Traffic Scheduling System we have built. This includes 3 Components:

- **Controller –** Controller of Coordinator on the cloud instance to manage API Calls, Database Dumps and for Signalling when the Traffic Schedule System should activate.
- **Image Processing –** Turns the Traffic Image Data to a Traffic Density Matrix using python's OpenCV. It then gives this data back to the controller and also passes it on to the Scheduling ML Model
- **Scheduling ML Model –** Turns the Traffic Density Matrix to a Signal Schedule and gives it to the controller

6.4.2 Roles of the Controller Step by Step

- Traffic Signal Client sends a Heartbeat to the Controller
- Controller Marks the specific client as Online
- Controller reads pre-mapped coordinates of the specific junction
- Controller Does an API Call to the Data source with this coordinate as parameter
- If Data source fails to respond, the API request is made again as long as the client is online
- If Data source responds with Maps Image Data, the controller marks the Scheduling Processes as started for the particular junction (client)

- The Data is sent to the Traffic Image Processing algorithm
- If the Image processing algorithm doesn't give an output in some time, The API call is made again to get a new image.
- When the Image Processing algorithm gives the Traffic Density Matrix as output, it is sent to the controller and also directly to the Scheduling ML Model. This Traffic Density Matrix is saved temporarily by the controller.
- When the ML Model gives the Traffic Signal Schedule to the Controller, it is sent back to the Traffic Signal Client.
- The Traffic Matrix and the corresponding Schedule are both committed into a database by the Controller

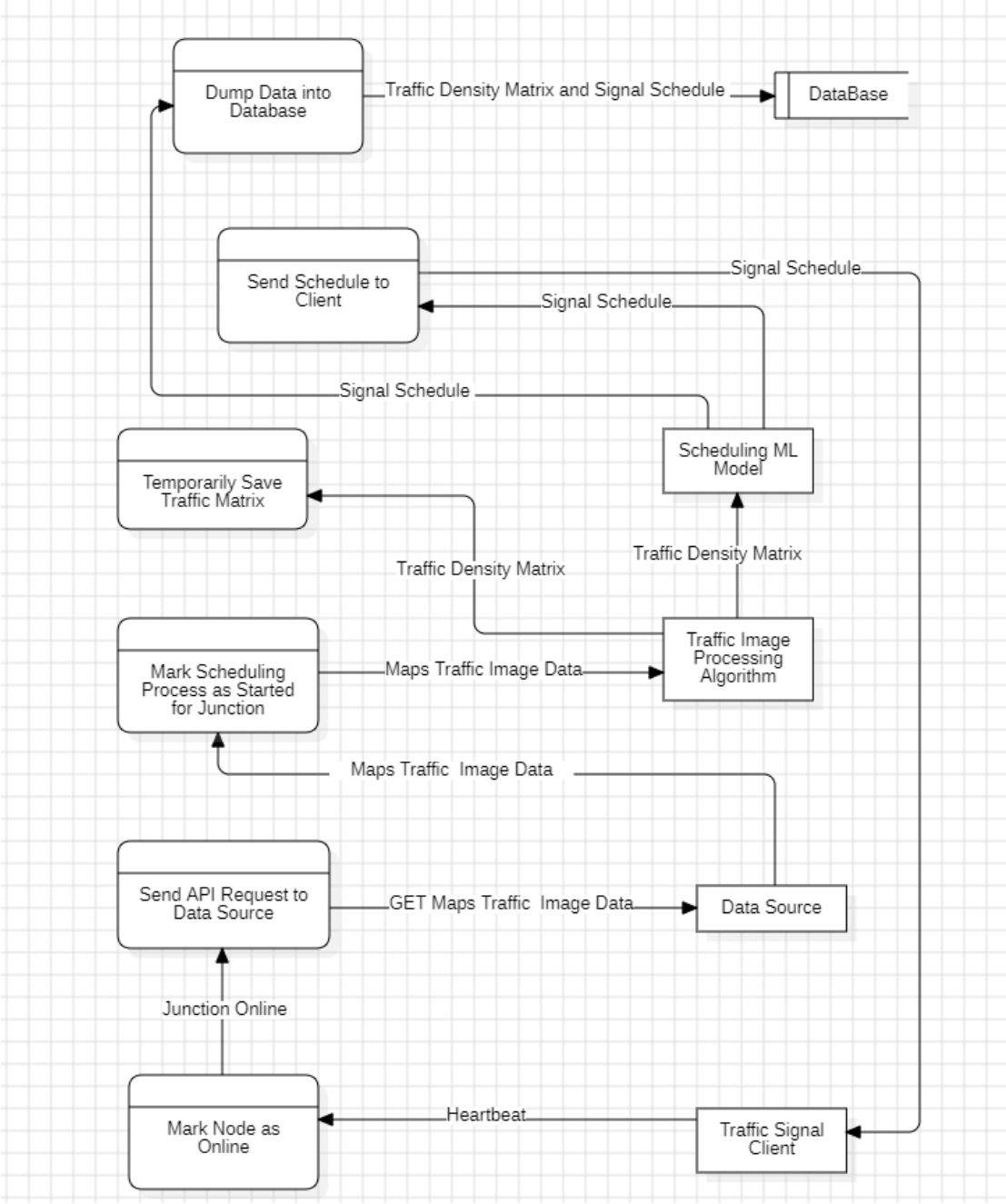

Figure 6.8 - Data Flow inside the Controller

6.5 Traffic Image Data Processing

We will be obtaining traffic data from google maps' API. The image will be zoomed in enough to be limited to only one 4 lane junction at a time. This is done for standardization purposes. The traffic in a given section is colour coded from green to dark red. In an incremental manner:

- Green- very low
- Yellow- medium
- Red- High
- Dark Red- Very High

Numeric representations to these colours will be assigned.

Upon obtaining the image, it will be logically slicing it into 3 concentric circles, with the radii being x, 2x, and 3x respectively. This will be done using image processing algorithms using python's OpenCV. With the traffic signal being at the centre.

These slices are to be converted into a 2d array[i][j], where i represents the road with respect to the signal, i.e., top, right, left, down, which are indexed to i = 0,1,2,3 respectively. j represents the circle, with j=0 being the innermost circle in any lane, and j=2 the outermost.

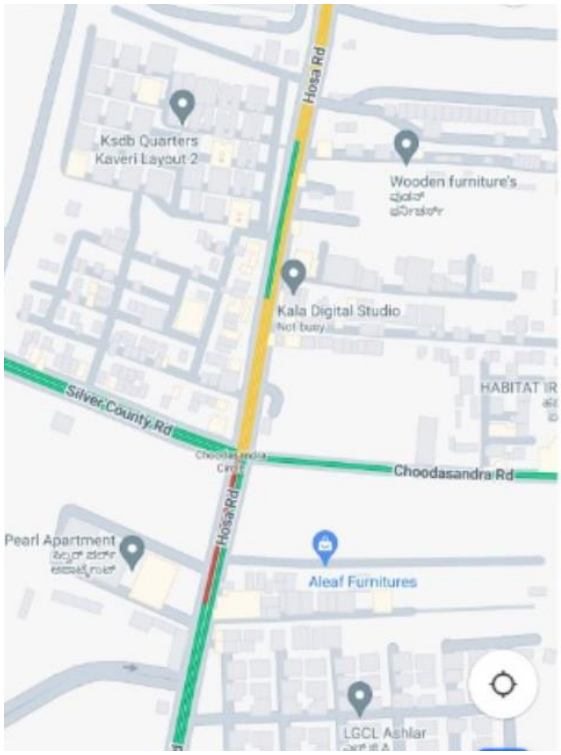

Figure 6.9 – 2nd Sample Image of Datapoint used

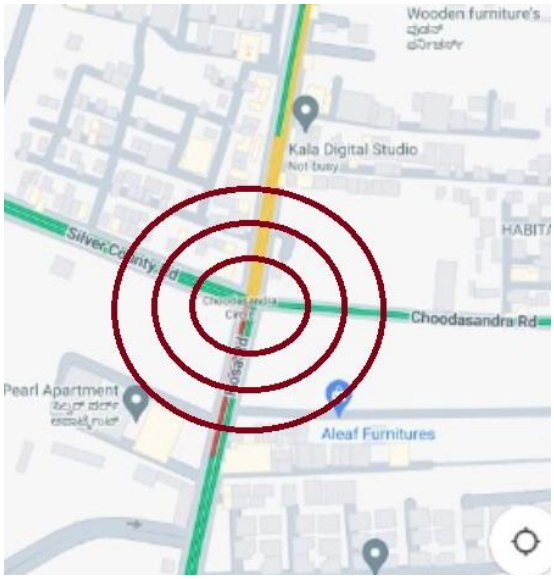

Figure 6.10 - How the Image Processing code works

This array is the input to our SVM model.

6.6 Signal Scheduling Machine Learning Model

A model using SVM is made which replicates the current existing traffic control system and tries to optimize it using only classifications.

The input to this model will be the matrices generated by the image data processing algorithms. Our model uses the generated matrix and classifies them to predefined classes using SVM. The output of this model will be an array which gives us the predicted traffic timings for our current scenario.

This model has its limitations as doing this for real time data will be very difficult and the efficiency will reduce.

The SVM based model is built as a prototype model for our main model. To overcome the above limitation we are building , DTSA(Dynamic Traffic Scheduling Algorithm). This algorithm takes the same matrix as input, these matrices represent the real time traffic data.

- This DTSA model consists of an LSTM model which is used for complex scenarios and an SVM model which takes care of the more simple scenarios as previously discussed.
- LSTM will work together with SVM to enhance the classification process.
- LSTM works well on sequential data and does the feature extraction which is then given to the SVM to get a classification.
- There will be changes done to the models as per the requirements, as real time data is unpredictable.

These models will work together to enhance the classifications. The efficiency will also increase with DTSA compared to the prototype. DTSA model works with historic data for training as the training for LSTM takes a lot of data. This model mainly focuses on real time classification and providing suitable timings for each of the lanes.

6.7 Simulation Software for testing purposes

6.7.1 SUMO

The simulation tool we will be using in our project is SUMO- Simulation of Urban Mobility. It's an open-source software which is used for simulating the road network topology, traffic conditions and signals in all urban and suburban areas.

SUMO provides detailed virtual environments that simulates real world traffic flow scenarios, which helps us in experimenting with new and different traffic management strategies to achieve efficient traffic flow conditions. This tool allows users to test traffic control strategies like signal timing, lane management, and adaptive signal control algorithms which would ultimately be reducing congestion, pollution and decreasing the rate at which road accidents take place.

SUMO also has tools and visualization features for analysis which helps us to observe and study different traffic patterns, travel times and movement of the vehicles. We can generate reports for better understanding of the behaviour of traffic and performance metrics

It supports various input and output formats for importing real-world traffic data, road network data, and traffic control plans, as well as exporting simulation results for further analysis.

6.7.2 Why Sumo is viable

- Open-Source and Free
- Comprehensive Features
- Active Community Support
- Scalability and Performance

Based on the above-mentioned advantages of SUMO, we decided it to be our simulation model for our dynamic traffic signal scheduling system. We plan on simulating a 4-lane junction with three lanes (left turn, right turn and straight) as seen in google maps which would also indicate the location of the traffic signal poles at the beginning of each lane.

The main aim is to incorporate the optimised signal timings we get from our machine learning model such that it simulates the traffic flow in SUMO.

6.7.3 How we incorporate it in our project

By default, SUMO uses static scheduling for traffic signals, which means it’s not changing in real time depending on congestion level or other real time factors but is having a predetermined time plan for all kinds of traffic conditions.

We through our project aim to resolve this limitation by making the signal timing change dynamically depending on the congestion levels in the lanes. To achieve this, we will be making use of TraCI (Traffic Control Interface) which is a third-party library for python which serves as an interface connecting python scripts to SUMO. This enables us to get real time information from the simulation and send commands which will control signal timings.

6.7.4 Detailed Approach and Data Flow

- The machine learning model we will be using will take the real time congestion levels in the lanes of the 4-way junction.
- Based on this information, our machine learning model provides us with the optimised signal timings for that particular junction's signals.
- In our SUMO environment we have the same junction simulated with the location of the traffic signal poles and road network as per the image from google maps. It also has the congestion levels simulated.
- These optimised signal timings given by our model are then communicated to the SUMO simulation using TraCI.
- TraCI receives these optimised signal timings and applies them to the simulated traffic signals
- Based on the signal timings provided by TraCI and SUMO we can observe the traffic flow in the simulation and also do the required analysis on how it's better than the static scheduling we have at present.

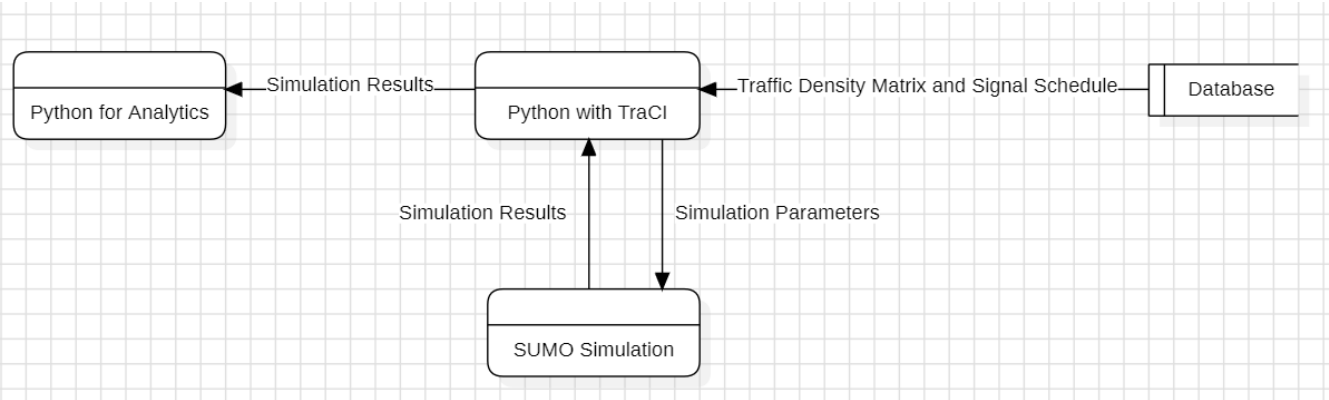

Figure 6.11 - Using TraCI to Connect with SUMO

CHAPTER VII

VII. CONCLUSION OF CAPSTONE PROJECT PHASE–1

With the amount of work done in our project at the moment, we have completed around 20% of the project.

1) We Conducted extensive research and analysis to determine the technologies and methodologies that will be utilized in various phases of our project.

2) Progress has been made in acquiring a source for data. Namely, the google maps API.

3) Our team has successfully completed the preliminary requirements specification (PRS) and high-level design (HLD), which will play an important role in the overall project process. The architecture of the project has also been finalized.

4) We have found supporting evidence through research for the employment of SVM model, although it comes with certain limitations.

5) To overcome some of these limitations, we aim to integrate an LSTM model in the project.

6) We have finalized on the use of SUMO for simulation. This will be employed in the testing phase and will be used to simulate traffic.

CHAPTER VIII

VIII. PLAN OF WORK FOR CAPSTONE PROJECT PHASE-2

With the completion of the documentation, design phases, and basic collection of the dataset for the application, we have successfully completed phase 1. In phase 2, our focus will be on setting up the architecture, building the model, implementing, testing and refining the model.

This can be done in steps as follows:

- Acquire the dataset after finalizing the 4-lane junction that will be considered for this project.
- Complete the image processing model to refine this dataset and get the matrices for the next part.
- Complete the prototype model, this involves refining the SVM part of our model and finalizing the classes.
- Implementing the DTSA model and refining the LSTM for feature extraction and getting better classes for SVM.
- Simulate using SUMO and test performance with the cloud architecture in place.