CS531- Python Applications Programming

Traffic Governance System

Submitted By:
Akshay Patel (19938) Feng Guo(19896)
Prince Paneliya (19885) Rakesh Banda (20048)

Guided By:
Prof. Dr. Zheng Lei

Problems: Inefficient Traffic Management

- **Inefficient Signal Timing:** Fixed signal cycles don't account for changing traffic patterns.
- **Traffic Imbalance:** Congestion builds in high-volume lanes while other directions have long green lights despite low traffic.
- **Long Wait Times:** Drivers experience excessive delays, especially during peak hours.
- **Environmental Impact:** Stop-and-go traffic increases emissions and pollution.

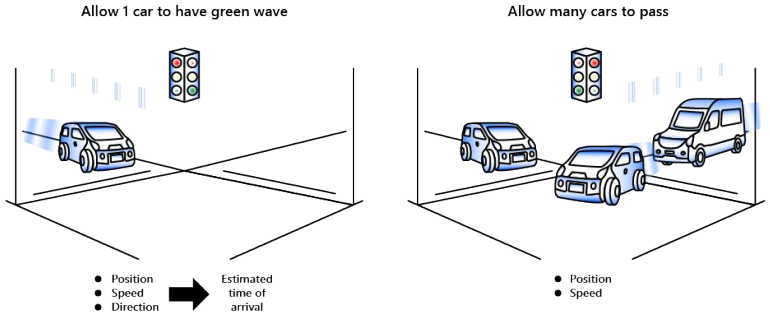
Introduction: Traffic Governance System

Definition:

Traffic Governance System is a system where centrally-controlled traffic signals and sensors regulate the flow of traffic through the city in response to demand.

How does it work:

- Smart traffic governance includes crowd detection at traffic signals.
- It includes video analysis. It detects the lane with heavy traffic.
- A notification is then sent to the traffic police booth at that particular signal and the signal timer is then altered accordingly.
- The increase in signal timer will allow more vehicles to move at a time. This reduces waiting time of people at traffic signals and also reduces traffic congestion at crossroads.
- it also reduces pollution and saves resources

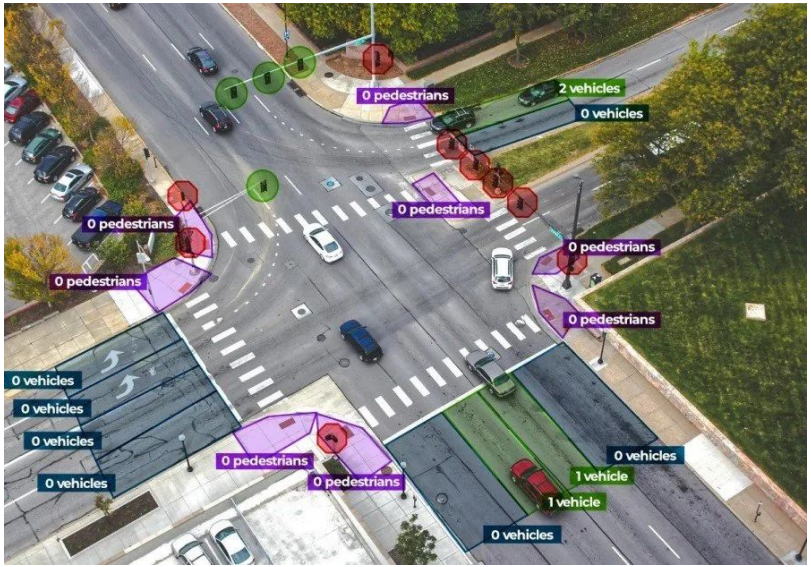


Proposed Solution: Dynamic, Sensor-Driven Control

Our solution: A software simulation environment enables us to design, test, and refine adaptive control policies.

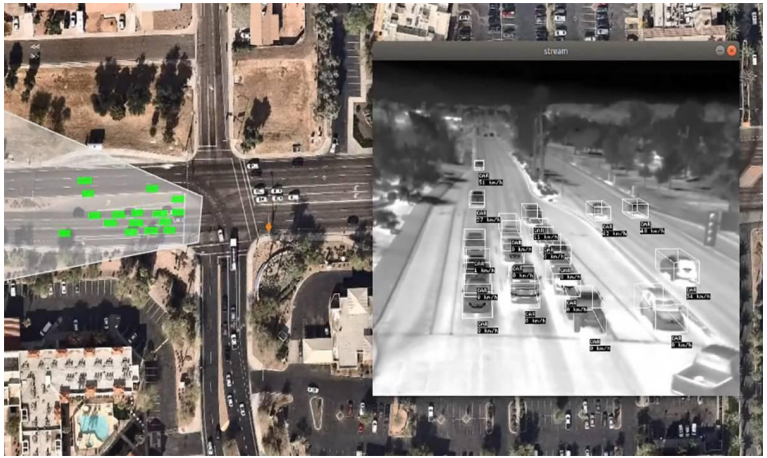
Key Elements:

- **Realistic Traffic Simulation:** Mimics real-world intersections and signals
- **Sensor Data Integration:** Diverse 'sensor' data for rich decision-making
- **Adaptable Policies:** Quickly test and compare traffic control strategies
- **Key Goal:** Finding the most efficient ways to reduce wait times and congestion



Key Features and Technologies

- Advanced technologies: image processing, computer vision, AI, intelligent controls
- Real-time data collection via sensors and communication systems
- Priority-based decisions and dynamic signal timing adjustments
- Algorithms for traffic routing and optimization based on current conditions



Potential Alternative Approaches

- Fixed Timers
- Adaptive Traffic Control Systems (ATCS)
- Traffic Prediction & Routing
- Intelligent Transportation Systems (ITS)
- Physical Upgrades (lanes, intersections)

Our Simulation-Driven Model (Advantages)

- **Real-time Optimization:** Adapts signals based on traffic, reducing congestion and improving flow.
- **Prioritizes Busy Lanes:** Gives more green time to congested lanes, reducing wait times.
- **Less Congestion, Faster Travel:** Minimizes congestion and optimizes signals for faster travel times.
- **Cleaner City, Safer Roads:** Reduces pollution and improves safety by smoothing traffic flow.

.

What We Built

Presentation showcasing our achievements in image processing and AI technology

Vehicle Tracking Module

Vehicle Counting

Provides real-time count of vehicles in traffic flow

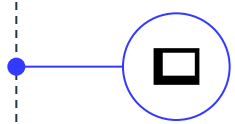
Accurate Tracking

Tracks vehicles with high precision and reliability

Traffic Videos

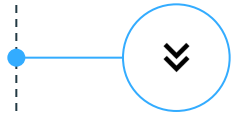
Specialized for analyzing traffic footage

Traffic Signal Optimization



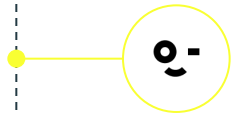
Rule-Based System

Algorithmic rules utilized to optimize signal timing.



Traffic Data

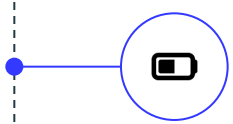
Data analysis of traffic patterns for improved signal changes.



Timing Adjustment

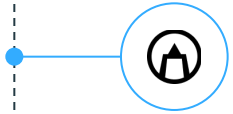
Real-time adjustments made based on current traffic conditions.

Video Upload Tool



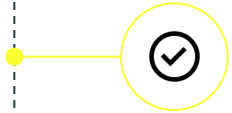
Optimized Signal

Tool generates optimal signal timing for uploaded videos



Timing Results

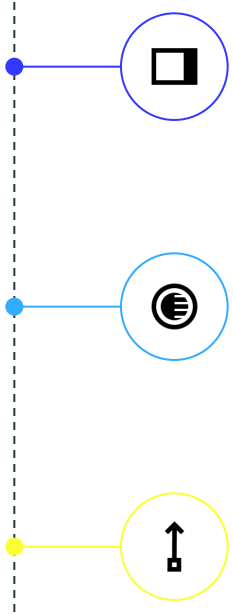
Results ensure best quality and performance for video signals



Web Interface

User-friendly web interface for easy video uploading

Traffic Data Analysis



Analytics

Utilize data visualization tools for in-depth analysis

Visualization

Generate actionable insights for traffic management

Insights

Customize user interface for user-friendly experience

Object Detection Implementation:

Object Detection and Tracking:

- Implemented object detection using YOLOv3 algorithm to detect vehicles (cars, trucks, buses) and other objects (people, bicycles) in traffic videos
- Used pre-trained weights and COCO dataset labels for robust object detection
- Set a confidence threshold of 0.6 to filter out weak predictions
- Leveraged dlib's correlation tracker to assign unique IDs to detected objects and avoid duplicate counting

Efficient Data Collection:

- Converted video input into individual frames using OpenCV
- Each frame was processed by the YOLOv3 object detection model to count the number of objects
- Tracked unique objects across frames to maintain accurate counts

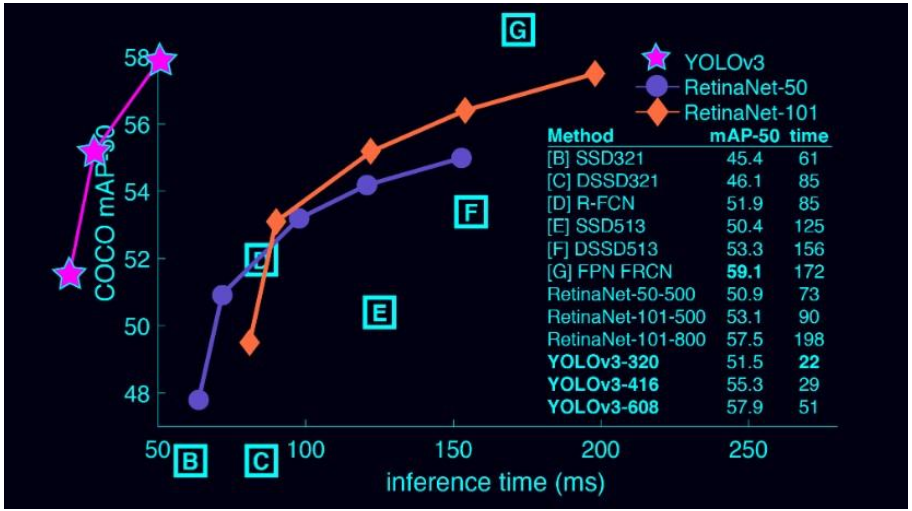
Overcoming Challenges:

- Learned about correlation trackers from an IEEE paper on Vehicle Counting for Traffic Management System
- Implemented the correlation tracker to address the issue of duplicate object detection

Why YOLO (you only look once) ?

YOLO vs Other deep learning techniques

- YOLOv3 is extremely fast and accurate. In mAP measured at .5 IOU YOLOv3 is on par with Focal Loss but about 4x faster.
- Moreover, you can easily tradeoff between speed and accuracy simply by changing the size of the model, no retraining required!
- Recent advancements in machine-vision algorithms and high performance computing have improved image classification accuracy to a great extent. In this Project, YOLO models- deep learning techniques were used to detect traffic congestion from camera video



YOLO vs DCNN and SVM

YOLO vs Other deep learning techniques

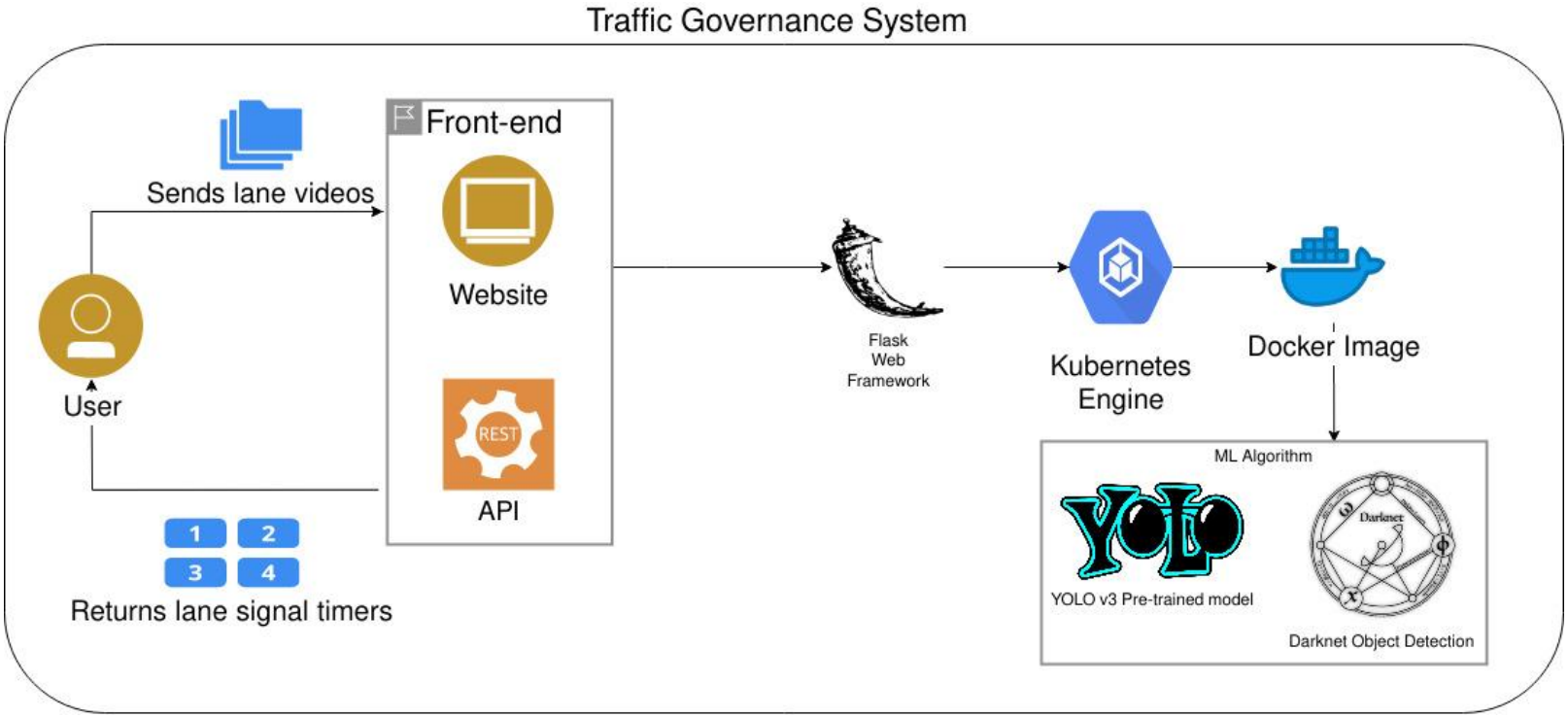
- YOLO and DCCN(Deep Convolution Neural Network) achieved 91.5% and 90.2% accuracy, respectively, whereas SVM's(Support Vector Machine) accuracy was 85.2%.
- Precision, recall and accuracy values obtained from the three algorithms: Where YOLO achieved the highest precision, recall, and accuracy

Method	Precision (%)	Recall (%)	Accuracy (%)
YOLO	88.6	94.3	91.4
DCNN	86.9	93.9	90.2
SVM	82.8	88.5	85.7

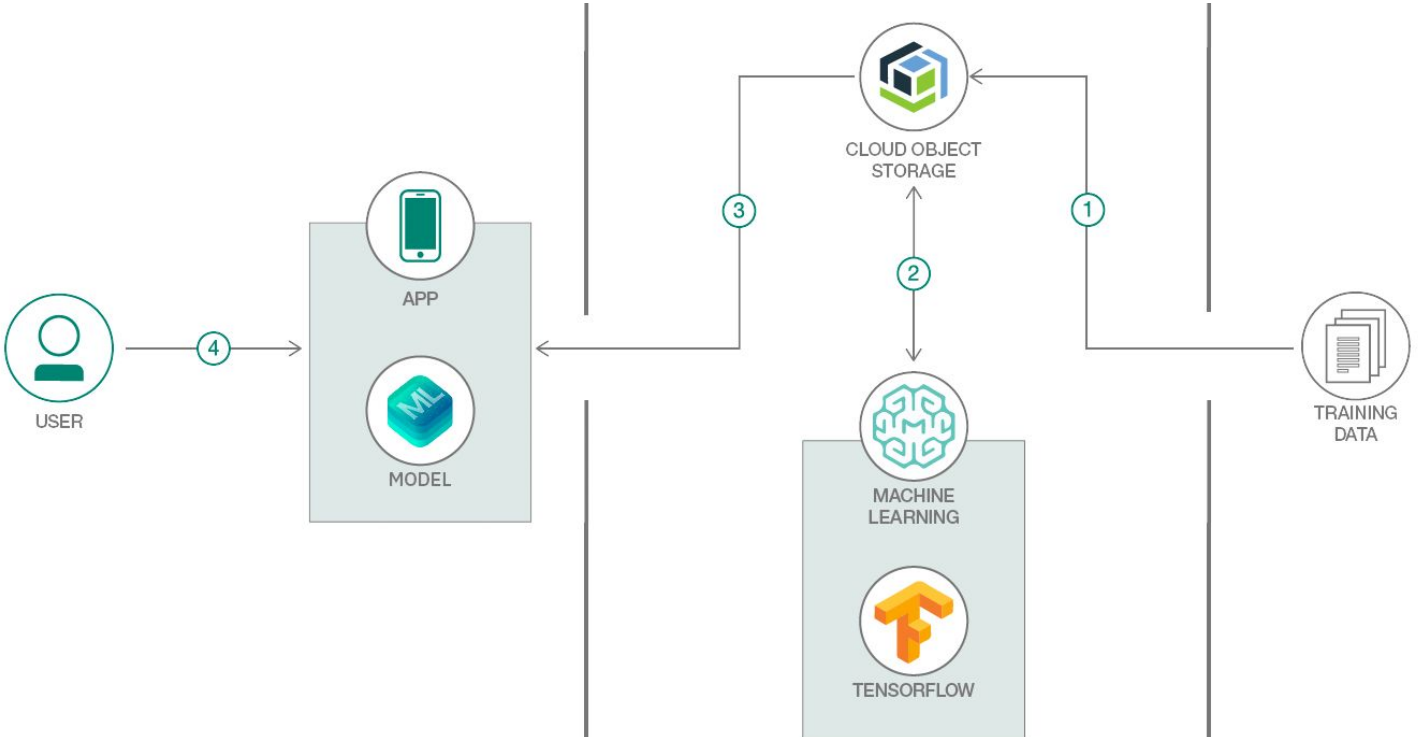
Kubernetes

- Since the algorithm consumes atleast 3GB of memory and needs atleast 4 CPUs to execute at a moderate pace, it was tough to get such processing power in free tiers in Cloud Platforms. Hence, we installed minikube to create a local Kubernetes cluster and virtualbox to create VM to simulate Kubernetes Engine.
- ***Note*:** By trimming the video, we were able to utilize IBM Kubernetes resources to host the docker image and run the application. IBM Kubernetes creates a free Cluster with 2 vCPUs and 4GB RAM and allows NodePort service to communicate with the pods.

Architecture Diagram



Architecture Diagram



System Features

User: User can view notifications sent by the admin and take necessary actions.

Admin: The admin can detect traffic at crossroads by video analysis.

Database: Database stores the data and performs analysis on the data to generate the required result.

WebPage: Upload Video and get the timing of the signal for the each video

System Features

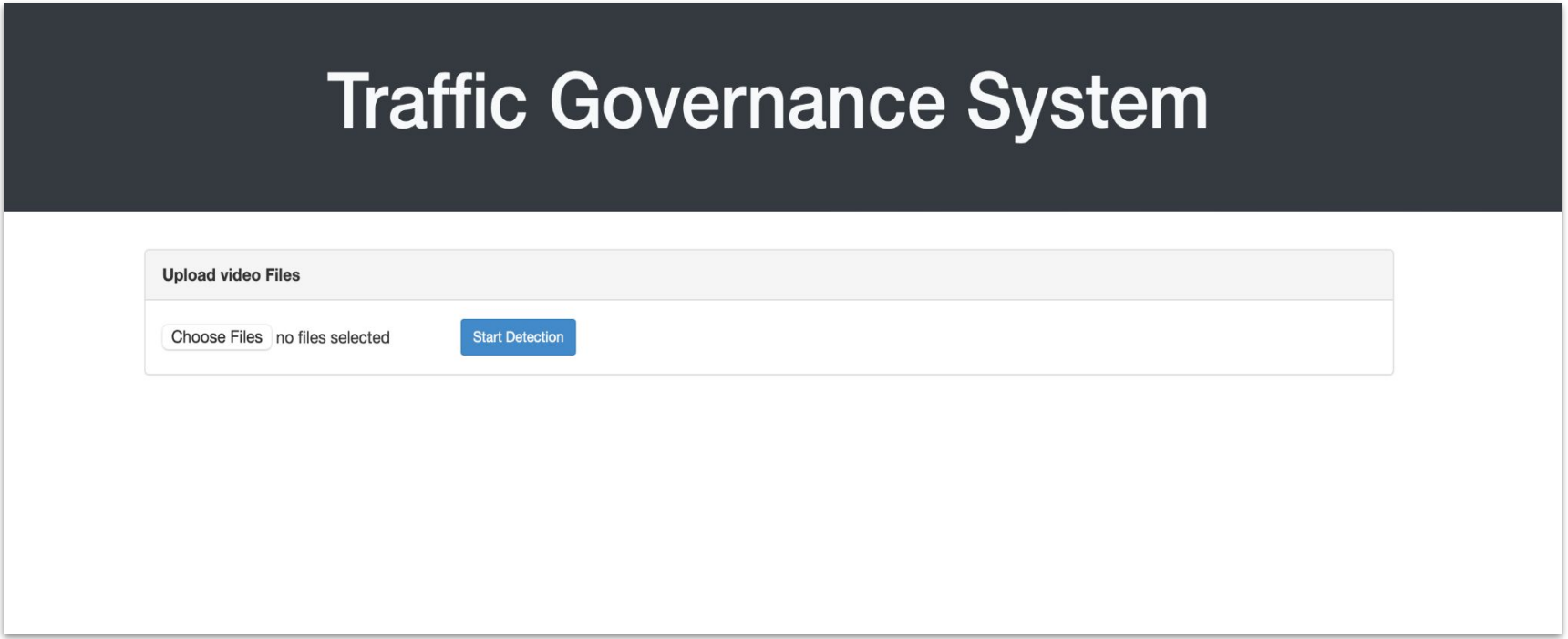
User: User can view notifications sent by the admin and take necessary actions.

Admin: The admin can detect traffic at crossroads by video analysis.

Database: Database stores the data and performs analysis on the data to generate the required result.

WebPage: Upload Video and get the timing of the signal for the each video

Traffic Governance System



Design and Implementation

The language that is used for coding is Python, HTML, CSS For working on coding phase.

We will make use of Pycharm for working. Also, we will make use of the online references.

We will make use of the existing Python libraries such as numpy, imutils, dlib, OpenCV and Flask.

requirements.txt

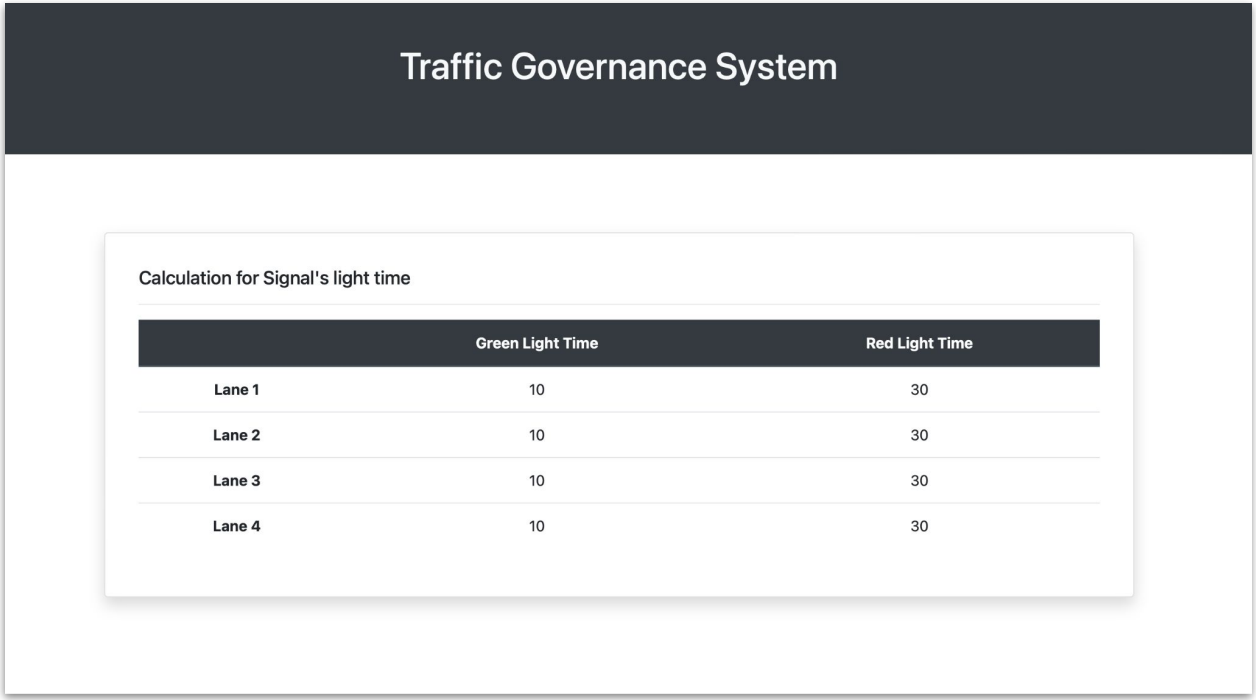
requirements.txt

```
1  Flask
2  imutils
3  numpy
4  opencv-python
5  dlib
6
```

User Interface

Road Side	Green Signal (Seconds)	Red Signal (Seconds)
Side 1	10	70
Side 2	20	60
Side 3	10	60
Side 4	30	50

Final User Interface



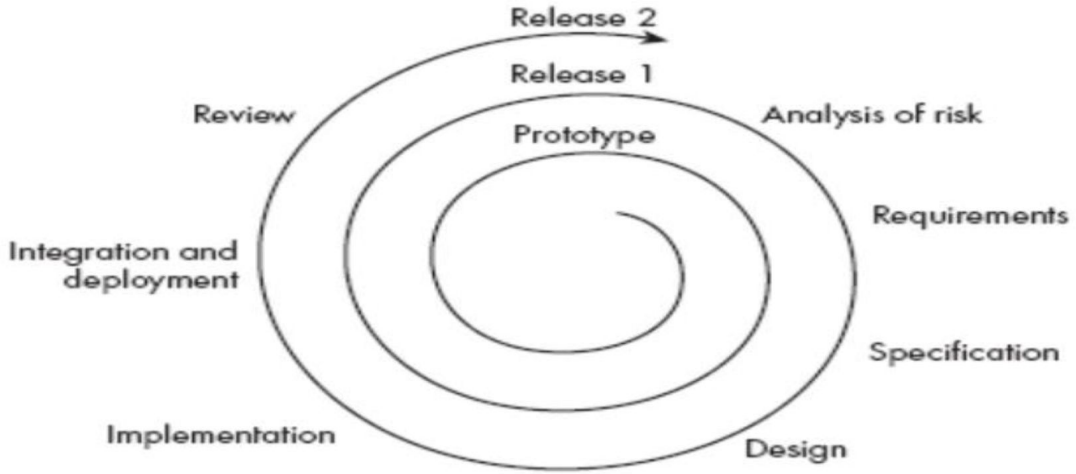
```
a = showDetection(files_in_dir[0])
print("🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥 ~ files_in_dir[0]:", files_in_dir[0])
print(a)
if a <= 10:
    green_light = 10
    print("this is signal timer for lane1 {}".format(green_light))
    lane1.clear()
    lane1.append(green_light)
    green_light_list.append(green_light)
elif a <= 20:
    green_light = 15
    print("this is signal timer for lane1 {}".format(green_light))
    lane1.clear()
    lane1.append(green_light)
    green_light_list.append(green_light)
elif a <= 30:
    green_light = 25
    print("this is signal timer for lane1 {}".format(green_light))
    lane1.clear()
    lane1.append(green_light)
    green_light_list.append(green_light)
else:
    green_light = 30
    print("this is signal timer for lane1 {}".format(green_light))
    lane1.clear()
    lane1.append(green_light)
    green_light_list.append(green_light)
```

```
car = 0
truck = 0
person = 0
bus = 0
motorbike = 0
bicycle = 0
# read the next frame from the file
(grabbed, frame) = vs.read()
```

```
            if LABELS[classID] == "car":
                car += 1
            elif LABELS[classID] == "truck":
                truck += 1
            elif LABELS[classID] == "person":
                person += 1
            elif LABELS[classID] == "bus":
                bus += 1
            elif LABELS[classID] == "bicycle":
                bicycle += 1
            else:
                motorbike += 1

if maximumTotal[0] > (bicycle + car + truck + person + bus + motorbike):
    maximumTotal[0] = maximumTotal[0]
else:
    maximumTotal[0] = bicycle + car + truck + person + bus + motorbike
```

Development Approach



SPIRAL MODEL

Delving into behind-the-scenes

MySQL

- Performance, reliability and ease of use.
- Best choice for the web based applications.
- A popular choice for embedded database and distributed among ISVs and OEMs.

Delving into behind-the-scenes

API : Flask

- Python flask for API design as flask supports extensions that can add application features as if they were implemented in the flask itself and doesn't require complicated boilerplate code
- Simple to use and easy to understand
- Ease of deployment on various platforms such as cloud and web servers

Delving into behind-the-scenes

Application Server : WSGI (Web Server Gateway Interface)

- Speedy and reliable server for running Python web applications
- Able to handle high traffic loads
- To forward requests from web server to python web application.

Future Expansion and Differentiation

- **Networked Control:** Coordinate simulations of multiple intersections across a city grid to optimize regional traffic flow. This enables intelligent traffic routing to reduce bottlenecks and improve overall efficiency.
- **AI-Driven Prediction:** Harness historical traffic data and machine learning to accurately forecast congestion hotspots. This allows for proactive traffic signal adjustments, preventing problems before they arise.
- **Unique Advantage:** Our simulator's adaptability allows rapid testing and integration of new technologies (advanced sensors, vehicle-to-infrastructure communication, etc.). This positions us at the forefront of traffic control innovation.

Impact on the community

- **Traffic Flow Optimization**: use real-time data from sensors, cameras, and other sources to monitor traffic conditions.
- **Cost Savings and Efficiency:** reduced fuel consumption, shorter travel times, and lower maintenance costs for road infrastructure.
- **Data-Driven Decision Making:** by analyzing traffic patterns, usage trends, and incident data, transportation agencies can make data-driven decisions to optimize infrastructure investments, plan for future transportation needs, and implement targeted improvements to specific roadways or intersections.
- **Environmental Benefits:** a positive impact on air quality and environmental sustainability
- **Improved Public Transit Integration:** Smart traffic management systems can integrate with public transit systems to provide priority for buses, trams, and other forms of public transportation.
- **Enhanced Safety:** nhanced road safety by implementing features such as adaptive traffic signals, real-time monitoring of traffic incidents, and integration with emergency response systems.

Major Accomplishments: What We Built

- **Core Simulation Framework:** Modeled traffic light behavior and generated realistic traffic scenarios
- **Image Processing Success:** Developed vehicle counting and tracking module for real-time data collection
- **AI Decision Engine:** Implemented rule-based system to optimize signal timings based on traffic conditions
- **Web Interface:** Built a platform for users to upload videos and receive optimized signal timing results

References

https://www.w3schools.com/python/
https://www.tutorialspoint.com/artificial_intelligence_with_python/index.htm
https://www.python-course.eu/python3_course.php
https://www.computer.org/csdl/proceedings-article/afips/1969/50730529/12OmNzC5TqA
https://www.fullstackpython.com/flask.html
https://numpy.org/doc/stable/reference/index.html#reference
https://pypi.org/project/imutils/#description
https://pypi.org/project/dlib/
https://pypi.org/project/opencv-python/
https://amsterdamsmartcity.com/projects/smart-traffic-management
https://www.bloomberg.com/news/articles/2013-02-05/how-virtual-traffic-lights-could-cut-down-on-congestion
https://www.linkedin.com/pulse/transforming-indias-urban-landscape-power-intelligent-traffic#:~:text=ITMS%20optimizes%20traffic%20flow%20through,travel%20times%20and%20fuel%20consumption.

?

Thank You