TraffiKAI:
An AI-Powered Solution for Efficient Traffic Management

(powered by oneAPI – oneDNN)

oneAPI

AGENDA

- Call-to-Action
- Problem Statement
- What data says?
- Objective
- Components
- Workflow
- Tech Stack
- Density Calculation
- Emergency Vehicle Detection
- Dynamic Traffic Signaling
- The oneAPI Edge
- Future Scope

TraffiKAI
A-EYE ON ROADS

CALL-TO-ACTION

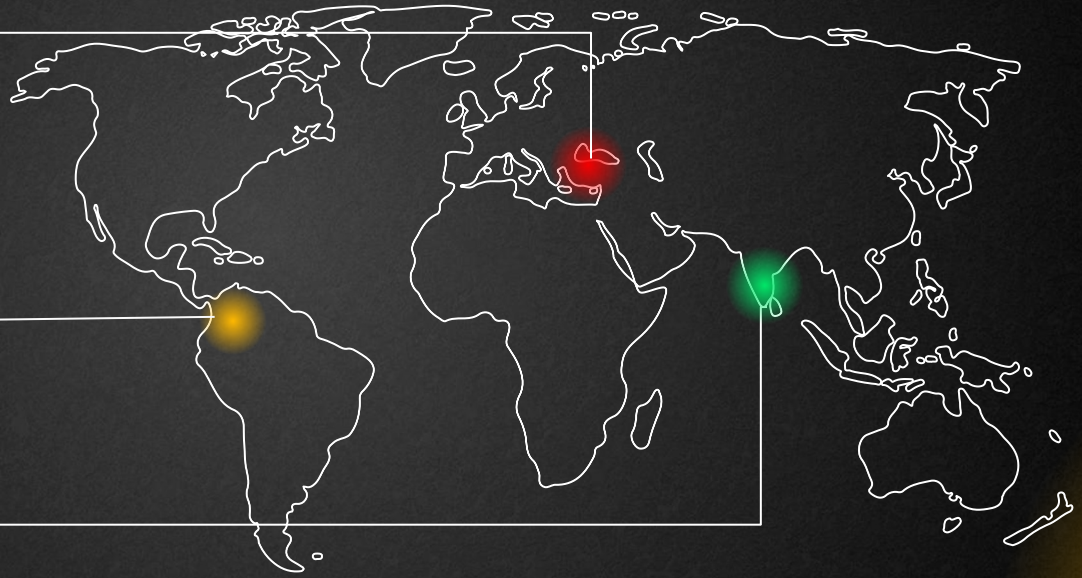
Ignite Change on the Roads:
Harness Intel® oneAPI's Power for AI-Enhanced Traffic Management – Implementing Dynamic Traffic Signaling and Instant Emergency Vehicle Detection, Shaping Safer, Smarter Cities Now.

PROBLEM STATEMENT

- The increasing number of vehicles in cities can cause high volume of traffic, and implies that traffic congestion has become more critical nowadays. Keeping the same infrastructure and making delta changes in the system using the power of AI & ML.

- Fatalities due to traffic delays of emergency vehicles such as ambulance & fire brigade is a huge problem. In daily life, we often see that emergency vehicles face difficulty in passing through traffic.

CITIES WITH THE WORST TRAFFIC IN THE WORLD



ISTANBUL

Ranked 1st

COLOMBIA

Ranked 4th

BANGALORE

Ranked 10th

DATA DOES NOT LIE!

- According to Times of India about 146,133 people were killed in road accidents in India in the year 2016. Unfortunately, about 30% of deaths are caused due to delayed ambulance.

- According to the Radhee Disaster and Education Foundation, one in 10 patients in India dies on the way to hospital.

- Another Indian government data shows that more than 50% of heart attack cases reach hospital late, which can constitute unavailability of ambulances too but majority of it is due to patients stuck in traffic.

OBJECTIVE

- Improve efficiency of existing traffic signaling system.

- Automate the traffic signal system and make it easy for the traffic police department to monitor.

COMPONENTS

- TraffiKAI is created by integrating multiple units, with a primary focus on incorporating Emergency Vehicle Detection and Dynamic Traffic Signaling.

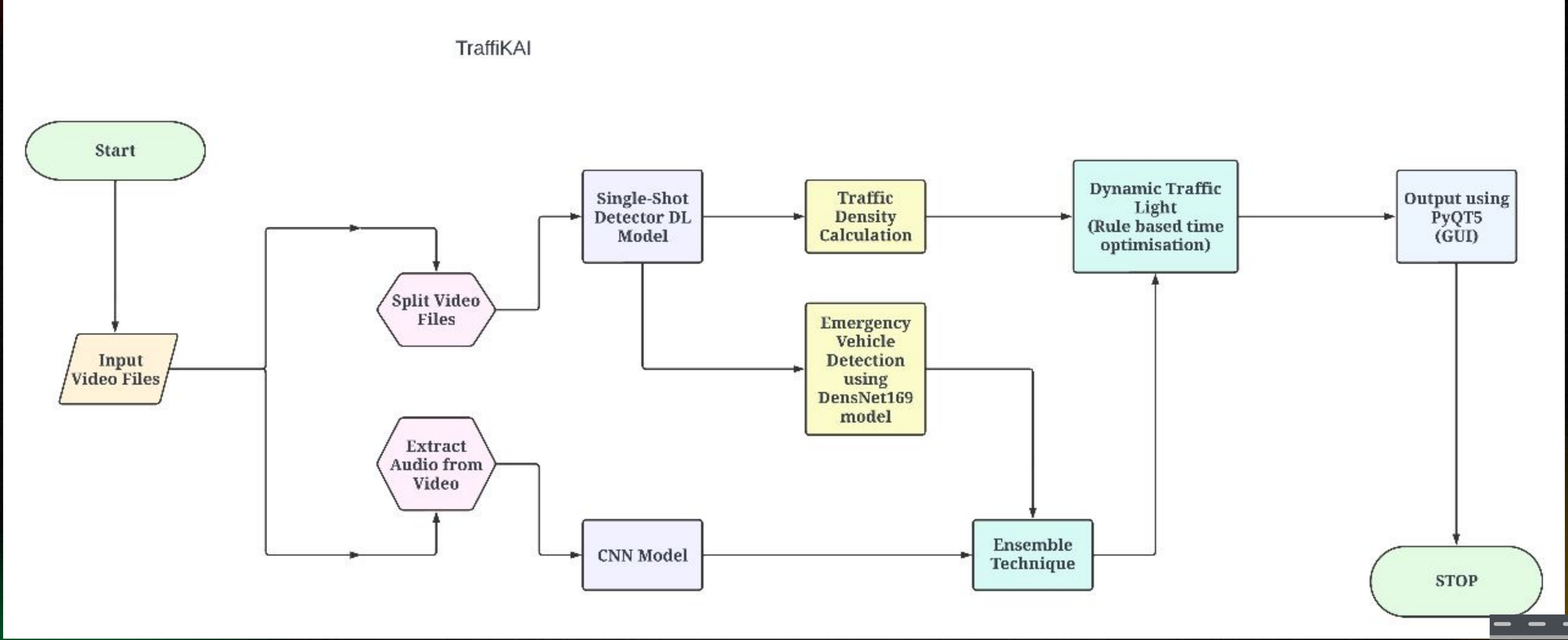
- The traffic density calculations plays a crucial role to process the number of vehicles based on the rule-based algorithm.

TECH STACK

Audio detection: Custom model using CNN
Video detection: DenseNet-169
Object detection: SSD (Coco Dataset)

GUI: PyQt5
Libraries: Tensorflow, OpenCV
Streamlit

SYSTEM WORKFLOW


TraffiKAI
Start
Input Video Files
Split Video Files
Extract Audio from Video
Single-Shot Detector DL Model
CNN Model
Traffic Density Calculation
Emergency Vehicle Detection using DensNet169 model
Ensemble Technique
Dynamic Traffic Light (Rule based time optimisation)
Output using PyQT5 (GUI)
STOP

TraffiKAI
A-EYE ON ROADS

DENSITY CALCULATION

```
if (best_boxes_scores[i][j] > 0.15) and (labels[int(best_boxes_classes[i][j])] in density.keys()):
            density_score[i] += density[labels[int(best_boxes_classes[i][j])]]
```

Where **best_boxes_scores** denotes the bounding boxes detected by the SSD model

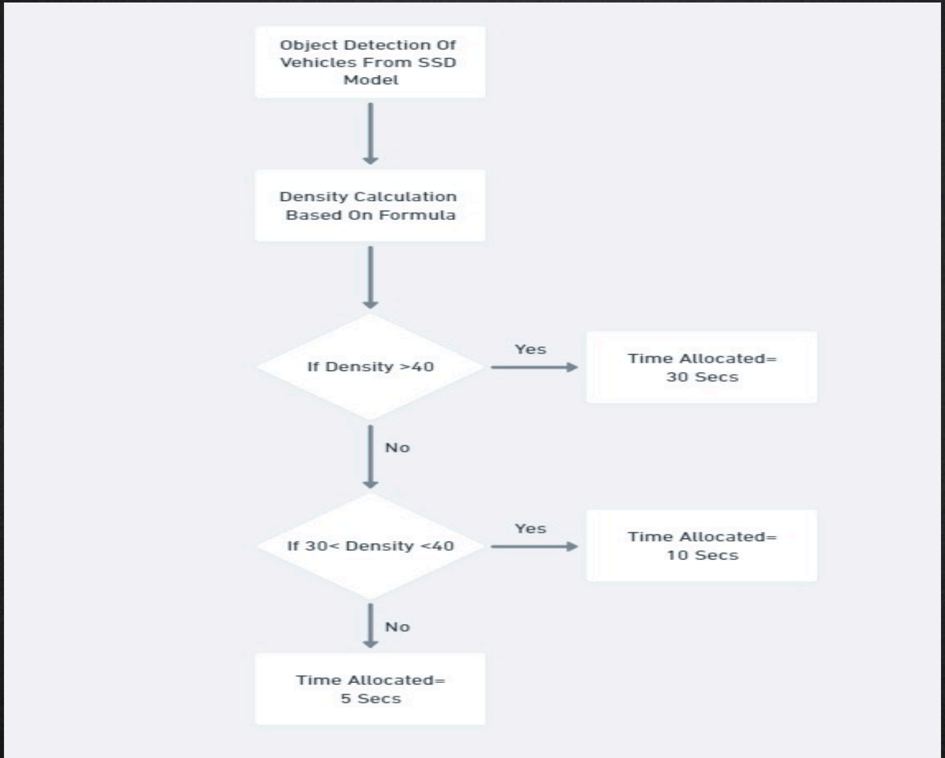
- The **best_boxes_scores** variable stores the scores of the bounding boxes detected by the SSD model.
- The **labels** variable stores a list of class names, where each class name is mapped to a class index.
- The **density** dictionary stores the density scores of each class.
- The **density_score** variable stores the density score of the current image.

EMERGENCY VEHICLE DETECTION

Ambulances and fire brigades require priority when navigating through traffic lanes.

- Emergency vehicle detection is done by ensembling both image classification models and audio siren detection models to improve accuracy.
- If the ensemble model predicts a probability of 0.90 or Above is a guarantee that an emergency vehicle is present.

DYNAMIC TRAFFIC SIGNALING



Object Detection Of Vehicles From SSD Model

Density Calculation Based On Formula

If Density >40

Yes

Time Allocated= 30 Secs

No

If 30< Density <40

Yes

Time Allocated= 10 Secs

No

Time Allocated= 5 Secs

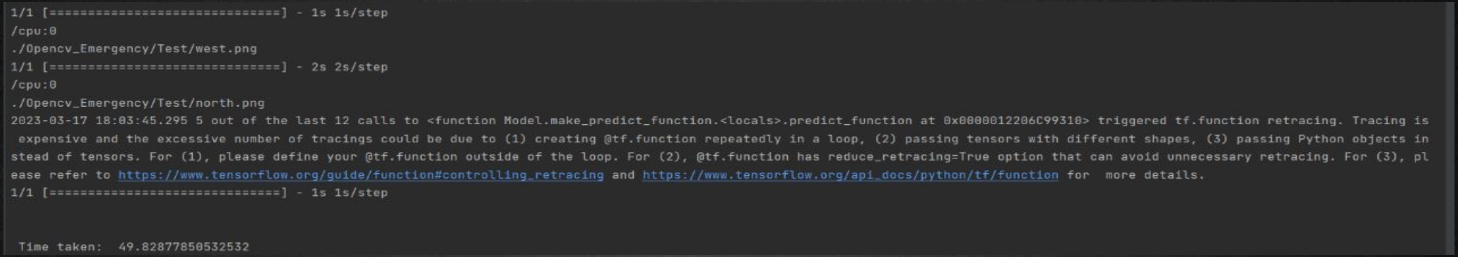
The oneAPI Edge

Tools used: Intel® AI Analytics Toolkit (AI Kit), Intel® oneAPI Deep Neural Network Library (oneDNN)

- TraffiKAI uses state-of-the-art deep learning frameworks like PyTorch and TensorFlow which are optimized for the Intel architecture by the oneAPI platform.
- Intel® Extension for Scikit-Learn* is also enabled to improve the performance.
- The Intel® AI Analytics Toolkit also has support for a number of pre-trained models such as DenseNet-169, YOLOv3, LSTM (audio) which are used in TraffiKAI.
- Using the pre-trained models, transfer learning has been implemented on the Intel® DevCloud for oneAPI.
- Intel® Distribution of OpenVINO™ Toolkit is also used to boost the object detection models.

The oneAPI Edge

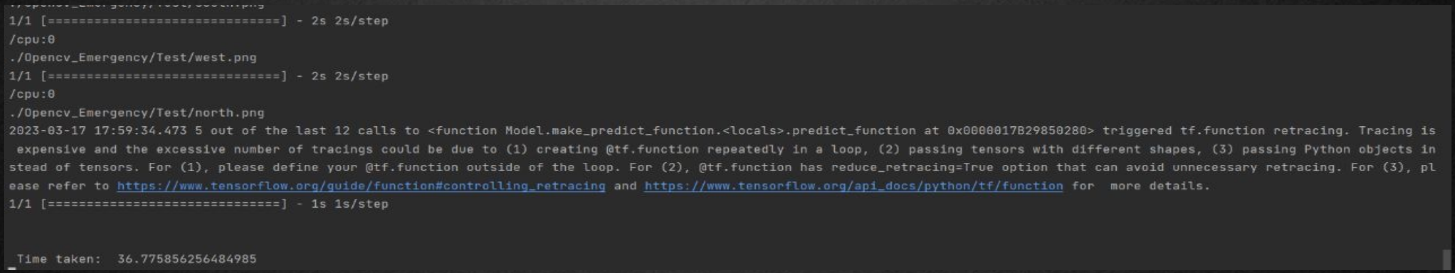
Time elapsed



```
1/1 [==============================] - 1s 1s/step
/cpu:0
./Opencv_Emergency/Test/west.png
1/1 [==============================] - 2s 2s/step
/cpu:0
./Opencv_Emergency/Test/north.png
2023-03-17 18:03:45.295 5 out of the last 12 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000012206C99310> triggered tf.function retracing. Tracing is
 expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects in
stead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), pl
ease refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/1 [==============================] - 1s 1s/step


 Time taken:  49.82877850532532
```

without oneAPI(base environment) : 49.82 seconds



```
1/1 [==============================] - 2s 2s/step
/cpu:0
./Opencv_Emergency/Test/west.png
1/1 [==============================] - 2s 2s/step
/cpu:0
./Opencv_Emergency/Test/north.png
2023-03-17 17:59:34.473 5 out of the last 12 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000017B29850280> triggered tf.function retracing. Tracing is
 expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects in
stead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), pl
ease refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/1 [==============================] - 1s 1s/step


 Time taken:  36.775856256484985
```

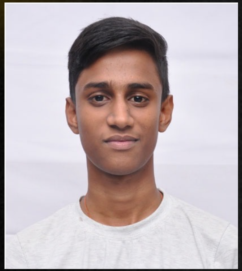
with oneAPI(oneapi environment) : 36.77 seconds

TraffiKAI
A-EYE ON ROADS

FUTURE SCOPE

- Collecting the data from junctions near hospitals allows better traffic management for easy emergency vehicle passing.
- Implementation of an automatic helmet violation detection system without any human intervention completes the system.

SOCIAL MEDIA HANDLES



MANIKANTA B

in https://www.linkedin.com/in/manikanta7342/

GURU AKHIL

in https://www.linkedin.com/in/guru-akhil-aa902a203/

NISHANK K S

in https://www.linkedin.com/in/nishank-satish-087304206/

VINEEL AKASH

in https://www.linkedin.com/in/vineel-akash-041670222/

SHREYAS K

in https://www.linkedin.com/in/shreyas-kuntnal/