Dragon-Fly: Vehicle Surveillance System

A Project Report

Submitted to the APJ Abdul Kalam Technological University

in partial fulfillment of requirements for the award of degree

Bachelor of Technology

in

Computer Science and Engineering

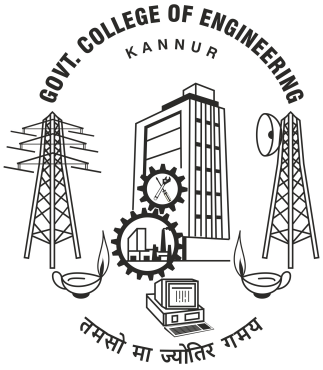
by

Abhinand C (KNR18CS005)

Edwin Jose George (KNR18CS027)

Lavanya E.V (KNR18CS032)

Shilpa Suresh (KNR18CS050)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

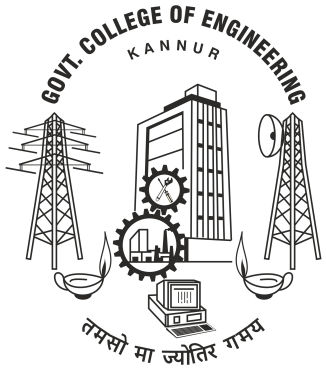
GOVERNMENT COLLEGE OF ENGINEERING KANNUR

KERALA

June 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GOVERNMENT COLLEGE OF ENGINEERING KANNUR

2021 - 22



CERTIFICATE

This is to certify that the report entitled **Dragon-Fly: Vehicle Surveillance System** submitted by **Abhinand C** (KNR18CS005), **Edwin Jose George** (KNR18CS027), **Lavanya E.V** (KNR18CS032) & **Shilpa Suresh** (KNR18CS050) to the APJ Abdul Kalam Technological University in partial fulfillment of the B.Tech. degree in Computer Science and Engineering is a bonafide record of the project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Dr. Rafeeque P.C
(Project Guide)
Professor
Dept.of CSE
Govt. College of Engineering
Kannur

Prof. Bincy Antony M
(Project Coordinator)
Assistant Professor
Dept.of CSE
Govt. College of Engineering
Kannur

Dr. Rafeeque P.C
(Head of Dept.)
Professor
Dept.of CSE
Govt. College of Engineering
Kannur

DECLARATION

We hereby declare that the project report **Dragon-Fly: Vehicle Surveillance System**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Dr. Rafeeque P.C

This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Abhinand C

Edwin Jose George

Lavanya E.V

Shilpa Suresh

Kannur

01-07-2022

Abstract

Vehicles are one of the most important means of transport. It have seen large improvements and innovation in the past decades. Almost all of our day-to-day activities are depended on the transport system. Hence there exist the need for safe policing of traffic. Existing method involves large human resource allocated for tedious task of tracking a vehicle. This project proposes an integrated system that make use of existing state-of-the-art technologies to support policing of traffic. The system incorporates the use of AI technologies to mimic the human intelligence for faster processing of tedious task. Major AI technologies mention in this project include YOLOv4 algorithm, DeepSORT tracker and Siamese Network. The system aims for successful re-identification of multiple vehicles across multiple camera viewpoints. The system provides interactive user interface that hides the complexities involved in achieving the task. Various aspects of security are also considered during the development of the project.

Abhinand C

Edwin Jose George

Lavanya E.V

Shilpa Suresh

Acknowledgment

We take this opportunity to express our deepest sense of gratitude and sincere thanks to everyone who helped us to complete this work successfully. We express our sincere thanks to Dr. Rafeeque P.C, Head of Department, Computer Science and Engineering, Government College of Engineering Kannur for providing us with all the necessary facilities and support.

We would like to express our sincere gratitude to the Prof. Bincy Antony M, department of Computer Science and Engineering, Government College of Engineering Kannur for the support and co-operation.

We would like to place on record our sincere gratitude to our project guide Dr. Rafeeque P.C, Professor, Computer Science and Engineering, Government College of Engineering Kannur for the guidance and mentorship throughout this work, providing with valuable timely suggestions and corrections.

We would also like to extend our gratitude to Dr. P Sooraj, Mechanical Dept. for inspiring us with the great idea and vision on our work. Thanks to Bineesh K.B sir, Dept. of CSE, for supporting us with technical resources. Thanks to Prof. Asjad Nabeel P for all those tips and tricks. Thanks to all the teaching/supporting staffs of Govt. College of Engineering Kannur for supporting and motivating us through the journey.

Finally I thank my family, and friends who contributed to the successful fulfillment of this project work.

Abhinand C

Edwin Jose George

Lavanya E.V

Shilpa Suresh

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Vehicles are one of the most important means of road transport for goods and services. Almost all the countries around the world have complicated road networks connecting remote locations. These road networks facilitate easy transport of people, goods, and services further encouraging the development of more networks. There has been exponential growth in the automobile industry to provide and market various automobiles into these road networks. This requires the need of policing this sector to ensure the safety of the people and the country. Controlling and managing this huge network is a great challenge due to its very vast and complicated network supporting a large population of vehicles. As part of policing, cameras are set up at various locations to monitor the traffic. However there exist the tedious work of manually analyzing the camera feed to detect and control the traffic. This project aims to tackle this particular issue.

1.1 Problem Statement

A novel approach to provide an integrated platform utilizing existing infrastructure to monitor transport sector, and extend timely information to policing agencies, with intuitive interface.

1.2 Background

Road traffic is one of the dynamic public sectors that involve huge human interventions. These include, but are not limited to construction of roads, transport of goods and other products, general public using public and private travel means, movement of construction goods, etc. The automobile is a booming industry, with a lot of companies producing new automobiles for the market meeting the need of the public. These include construction vehicles such as JCB, tractors, excavators, road rollers, trucks, etc; public goods vehicles such as vans, trucks, pickups; public vehicles such as buses, autos, taxis, etc; private vehicles such as cars, bikes, etc. Manufactures continuously supply new and modern automobiles to support the day-to-day activities of the public, and hence the nation.

This huge flow of vehicles demanded the setting up of rules and regulations for the safety of the people and properties. These rules ensured a smooth flow of traffic. However, at times these rules are broken by the public. To re-enforce these regulations proper policing is required. Officials are deployed at various points to physically inspect and regulate traffic. Upon finding any violation, the offender is fined. With the involvement of technology, speed cameras are set up on major roads to ensure that all vehicles do not travel beyond the safe speed limit. These cameras are special cameras that are triggered only upon finding a vehicle moving at high speed. When triggered they capture the license plate of the vehicle along with the time and place of the event. Obtaining license plate info helps to fetch associated vehicle information such as owner, address, vehicle make, model, type, etc. However, these are special cameras and a too costly to be deployed in larger and remote places.

At times, it was necessary to re-trace the route followed by a particular vehicle. As the technology grew, surveillance cameras are set up at various locations. Cameras allowed continuous monitoring of a particular section of the road. The feeds are saved at the severs and officials can replay the tape to extract the necessary information. These cameras are far less expensive than the speed cam, but also have lower resolution. These cameras find it difficult to extract unique features such as license plate information. They are mainly used for surveillance purposes with the just aim to record the event occurring in a fixed sector.

As the camera network was set up, the process of policing is still a tedious task. Officials need to manually replay each camera feed. Most of the time, the feed would be empty showing there was no traffic at that moment. Moreover, there would be many types of vehicles to consider. It is also possible for the vehicles to take alternative routes. The manual process of monitoring each feed is a highly time-consuming and tired-sum process. Moreover, it may need to identify a particular vehicle before it travels out of a particular zone.

This calls for the need for a system that could traverse the camera network detecting useful video frames and analyzing them to extract relevant information. This information can include features like type, color, model, location, time of the event, etc. A query can be issued to look up vehicles matching such descriptions. This saves a lot of time by skipping the dead frames. Another advantage is that, during an investigation, the witness can only provide such features to testimony an event. The suspect may be able to forge the license plate, but not the whole vehicle description.

1.3 Features

- **Traffic Analysis and Vehicle Tracking:** AI based traffic analysis, vehicle re-identification and tracking over the existing infrastructure.
- **Intuitive search:** A minimalistic and simple searching that even Naive users can utilize with ease, to trace vehicles.
- **Integration of any IP camera:** IP camera accessible through internet or local LAN or over VPN, can be integrated with ease.
- **Secured IP streaming:** Users can securely stream authorized IP camera without needing to access Camera Infrastructure Network using HLS over HTTPS.
- **Granular Role based permission management:** Enables controlled management of Camera Infrastructure with each fine-tuned permission management.

- **Map based CCTV visualization:** All IP cameras are configured with GPS coordinates, and other details, allowing easy visualization in Maps.

1.4 NVIDIA AI City Challenge

This project is one of the open challenges in NVIDIA AI city challenge, to perform Natural language-based vehicle track retrieval and City-scale multi-camera vehicle tracking. Various research and study are being conducted on this field.

Definition

AI City means applying AI to improve the efficiency of operations in city environments. This manifests itself in improving transportation outcomes by making traffic more efficient and making roads safer, improving building operations by making them more energy efficient, reducing friction in retail environments by speeding up traffic at retail checkout, etc. The common thread in all these diverse uses of AI is the extraction of actionable insights from a plethora of sensors through real-time streaming and batch analytics of the vast volume and flow of sensor data, such as those from cameras.

The AI City Challenge Workshop 2022

The AI City Challenge Workshop at CVPR 2022 specifically focus on problems in two domains where there is tremendous unlocked potential at the intersection of computer vision and artificial intelligence

1. The Intelligent Traffic Systems (ITS)
    - City-scale multi-camera vehicle tracking
    - Natural language-based vehicle track retrieval
    - Naturalistic driver data analytics

- Anomaly Detection

2. The Brick and Motar retail business
    - Automated checkout
    - Efficient Store utilization

1.5 Application

Mere processing of footage from camera have wide range of applications, especially in an dynamic sector like road transport. Due to urbanization and expansion of road networks, supported by the growth of technology and digitization, applications can vary from monitoring, planning to research, security and decision making. Some of the possible application of analyzing the traffic using camera network are:

- Identify and track the route taken by a targeted vehicle.
- Identify violations such as the check for helmet and passenger count in two wheeler, illegal turn taken at one-way roads, non-permitted vehicle entering the perimeter (eg: heavy trucks via old bridge is prohibited) etc.
- Identify events such as accidents or unexpected crowding, etc.
- Conduct research on the flow of traffic aiding for decision making for new road construction.
- Provide decision parameter for efficient routing of traffic to remove traffic blocks. Helps to change traffic load for emergency services such as ambulance, fire-trucks etc.
- Can be used to checkpoint vehicles, enhancing integrity and security of services. For example:
    - Confirming that cargo have left the station and is in-route to its destination.
    - Track and predict public transport services.

Chapter 2

Literature Review

Artificial Intelligence (AI) is one of the most researched topic in the domain of Computer Science. It is the artificial creation of human-like intelligence that can learn, perceive and process information. AI provides a powerful tool for solving image recognition, document classification as well as for the advancement of interdisciplinary problems. AI is an interdisciplinary domain that make use of various mathematical models, statistical models, neural networks etc to learn pattern and predict unseen data with comparable accuracy. Unlike traditional model, they learn the concepts and rules by learning from examples. Traditional model requires rules to be specified, that results in higher computation but higher accuracy. The ability of AI to generalize even accounting for noise with higher speed promoted various levels of growth, especially in the field of image processing.

Image processing is an interesting field of study that involves image classification, object detection, instance segmentation, object tracking etc. There have been various research and study [1] to overcome the traditional image processing that employ various algorithm using opencv. Various level of optimization was performed to build deep neural networks that can effectively extract relevant features to identify the object.

2.1 Vehicle Identification Systems

AI have always found it way in developing intelligent system in automobile industry. They are used for smarter manufacturing process, leading to design of efficient systems. They have also helped in business by recommending most efficient path in road network. They are also utilized to build intelligent autonomous vehicles that can react to changes in real-time environment and make way to its destination safely. AI is also used for monitoring purposes for enhancing security. Several parallel papers and researches are build developing such monitoring system using camera network. Below presented are few of such system that are deployed in selected area.

2.1.1 CityFlow

Urban traffic optimization using traffic cameras as sensors is driving the need to advance state-of-the-art multitarget multi-camera (MTMC) tracking. This work introduces CityFlow [2], a city-scale traffic camera dataset having the largest-scale dataset in terms of spatial coverage and the number of cameras/videos in an urban environment. The dataset contains a wide range of scenes, viewing angles, vehicle models, and urban traffic flow conditions. Camera geometry and calibration information are provided to aid spatio-temporal analysis. In addition, a subset of the benchmark is made available for the task of image-based vehicle re-identification (ReID). It also opens a way for new research problems such as vehicle pose estimation, viewpoint generation, etc.

It have compared and bench-marked several state-of-the-art models using Muiti-Target-Single-Camera and Multi-Target-Multi-Camera tracking. They have given emphasis on the visual-spacio-temporal reasoning to re-identification. This approach was able to product better and faster results, handle large scale recognition and is successful in conducting small predictions. However, they require multiple camera scanning same object at multi viewpoint at the same time. It also demanded high quality videos and was also set-up at junction points.

2.1.2 Vehicle Make and Model Recognition system

A Vehicle Make and Model Recognition (VMMR) system [3] can provide great value in terms of vehicle monitoring and identification based on vehicle appearance in addition to the vehicles attached license plate typical recognition. A VMMR system has a unique set of challenges and issues. Few of the challenges are image acquisition, variations in illuminations and weather, occlusions, shadows, reflections, large variety of vehicles, inter-class and intra-class similarities, addition/deletion of vehicles models over time, etc. The system extract image features from vehicle images and create feature vectors to represent the dataset. Then, two classification algorithms, Random Forest (RF) and Support Vector Machine (SVM) are used for classification. The proposed VMMR system recognizes vehicles on the basis of make, model, and generation (manufacturing years).

For feature extraction, HOG and GIST algorithm are utilized. These vectors are saved in database supporting faster processing speed and improved recognition accuracy, accounting for partial viewpoints. However they have limited region interest with high computation time with increase in block of HOG. The system only addresses local high dimensional features, and have no global representation.

2.1.3 Multi-level Feature extraction

The intelligent transportation system is currently an active research area, and vehicle re-identification (Re-Id) is a fundamental task to implement it. It determines whether the given vehicle image obtained from one camera has already appeared over a camera network or not. This task becomes more challenging because of intra-class similarity, viewpoint changes, and inconsistent environmental conditions. A system [4] is proposed which re-identifies a vehicle in two steps: first, shortlist the vehicle from a gallery set on the basis of appearance, and then in the second step, verify the shortlisted vehicle's license plates with a query image to identify the targeted vehicle.

The global channel extracts the feature vector from the whole vehicle image, and the local region channel extracts more discriminative and salient features from different regions. A

Siamese neural network is used to verify license plates to reach the exact vehicle. The system also account for color, model, and type as parameters of feature vectors. However, a significant impact of illumination can be found, which is worsen by background clutter. It also do not account for cross camera vehicle tracking.

2.1.4 Trends in Vehicle Re-Identification

Vehicle Re-identification (re-id) over surveillance camera network with non-overlapping field of view is an exciting and challenging task in intelligent transportation systems (ITS). Vehicle re-id matches targeted vehicle over non-overlapping views in multiple camera network. However, it becomes more difficult due to inter-class similarity, intra-class variability, viewpoint changes, and spatio-temporal uncertainty. In order to draw a detailed picture of vehicle re-id research, this research [5] provides a comprehensive description of the various vehicle re-id technologies, applicability, datasets, and a brief comparison of different methodologies. This research specifically focuses on vision-based vehicle re-id approaches, including vehicle appearance, license plate, and spatio-temporal characteristics. Some of the main technology taken under consideration are:

1. Magnetic Sensor-Based Vehicle Re-Identification
2. Inductive Loop-Based Vehicle Re-Identification
3. Global Positioning Systems-Based Vehicle Re-Identification
4. Vision-Based Vehicle Re-Identification
    (a) Feature Representation for Vehicle Re-Identification
    (b) Traditional Machine Learning-Based Vehicle Re-Identification
    (c) Similarity Metric for Vehicle Re-Identification
    (d) Fine-Grained Visual Recognition-Based Vehicle Re-Identification
    (e) View-Aware-Based Vehicle Re-Identification

(f) Generative Adversarial Network-Based Vehicle Re-Identification

(g) Attention Mechanism

(h) License Plate-Based Vehicle Re-Identification

5. Spatio-Temporal Cues-Based Vehicle Re-Identification Approaches

6. Hybrid Methods-Based Vehicle Re-Identification

2.2 Traditional Methods of Image processing

Traditional methods are computation intensive and hence slow. However, they have higher accuracy, but prone to noise. Some of the methods are:

1. Harris Algorithm

   Corner detection algorithm that identifies and compares the corners of objects and then compute the similarity.

2. SIFT: Scale Invariant Feature Transform

   Key-points of objects are first extracted from a set of reference images[1] and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors.

3. HOG: Histogram of Oriented Gradients

   It counts occurrences of gradient orientation in localized portions of an image, computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

4. GLOH: Gradient Location and Orientation Histogram

   It is a SIFT-like descriptor that considers more spatial regions for the histograms. An intermediate vector is computed from Location and orientation bins, for different dimensions.

5. Canny Edge detection

   It is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed.

6. SURF: Speed-ed Up Robust Features

   It is a fast and robust algorithm for local, similarity invariant representation and comparison of images. The main interest of the SURF approach lies in its fast computation of operators using box filters, thus enabling real-time applications such as tracking and object recognition.

7. DPM: Deformable Parts Model

   Special case of Dalal and Triggs Detector, making improvement over HOG and Support Vector Machine (SVM). Figure 2.1 shows the steps for detection of human.

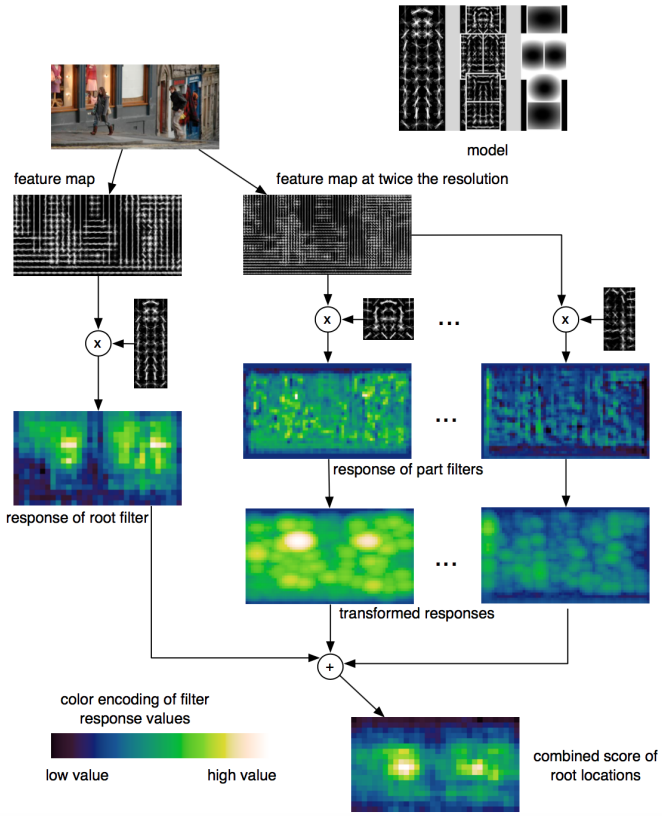
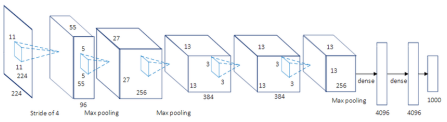


Figure 2.1: Deformable Parts Model for Human detection

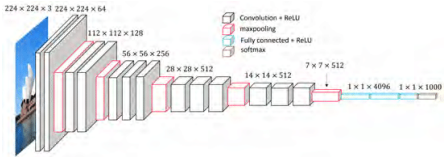
2.3 Neural Network Models

Neural Network model consists of neurons firing signals to change the output. Each neuron have weights associated along with activation function. The learning happens through the change of each weights. They require higher training time, but produce output faster along with the ability to resist the effect of noise. They generalize well event to identify occluded objects from image.

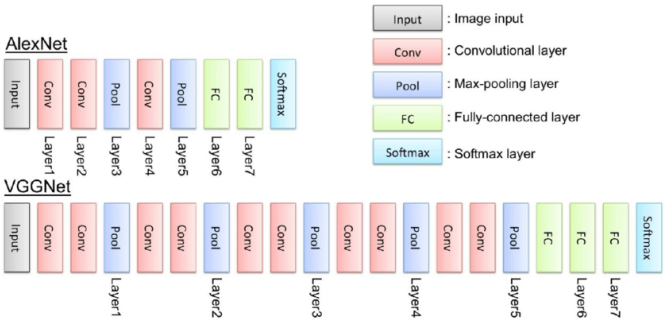
The major jump from traditional method into network model was on to the introduction of AlexNet Model. AlexNet have 5 convolution layers that effectively scale and extract the features. They are then pooled at each layers. At the end, a dense neural network gets trained to predict values for the output neuron, which are softmax-ed to obtain particular class. Dense neural network mimics a classifier. They are used for image classification. To account for various dimension of image, a concept of image pyramid is used to process image at various levels. VGG is an extension of AlexNet. Figure 2.2 shows the architecture of AlexNet and VGG model. Convolutional Neural Network (CNN) is a set of filters that extract a specific feature by applying some kernel.



(a) AlexNet Architecture



(b) VGG Architecture



(c) Layers of model. Classification done by FC layers

Figure 2.2: AlexNet & VGG Model

Owing to the success of AlexNet and VGG model, various study and research are conducted on the CNN layers to efficiently extract features. Some of the major contributions are summarized below.

Overfeat

- Problem addressed
  - CNN network can accept only fixed size images. To process a large image, the image needs to be cropped and resized to fit the dimension of the input of the network. The size constraint is imposed not because of the input layers, but due to the FC layers.
  - One solution involves manually cropping the image and feeding the network, a stimulation of sliding window. It involves redundant calculation at the image pixels.
- Solution Proposed
  - Implementation of FC as Conv network, removing the input constraint
  - Image pyramid of 6 sizes [461*569 , 425*497 , 386*461 , 317*389 , 281*317 , 245*245] creating spatial output dimensions. The spacial output dimensions show how many object can be detected from the image

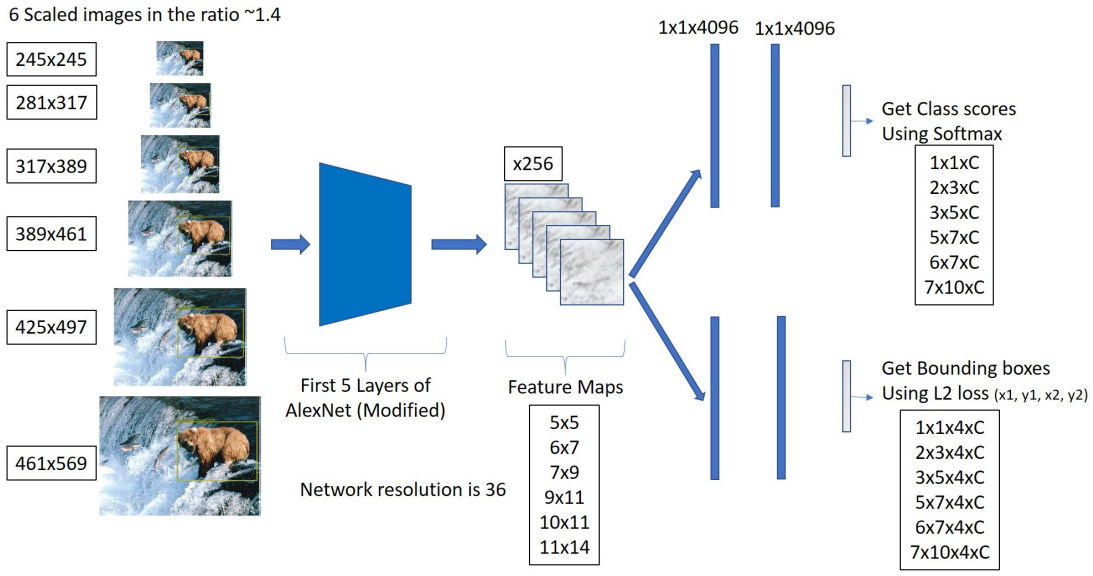


Figure 2.3: Overfeat Architecture

R-CNN: Regions with Convolutional Neural Networks

To bypass the problem of selecting a huge number of regions, we can use selective search to extract just 2000 regions from the image - called region proposals. These 2000 region proposals are generated using the selective search algorithm. These 2000 candidate region proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM to classify the presence of the object within that candidate region proposal. The algorithm also predicts four values which are offset values to increase the precision of the bounding box. Some of the drawbacks are

- It still takes a huge amount of time to train the network as we would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

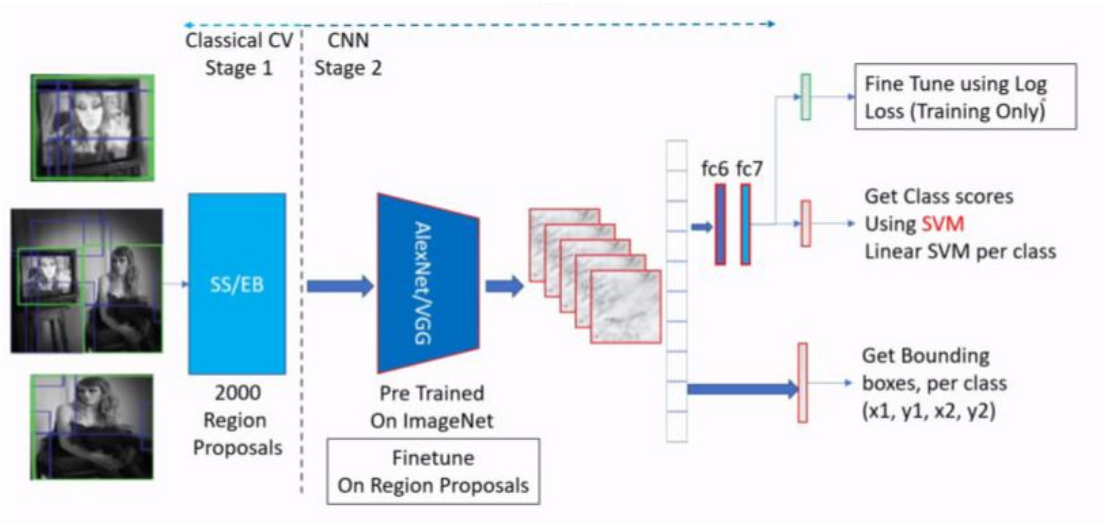


Figure 2.4: R-CNN model

The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called **Fast R-CNN**. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size. From the RoI feature vector, softmax layer is used to predict the class of the proposed region and also the offset values for the bounding box. "Fast R-CNN" is faster than R-CNN is because we do not need to feed 2000 region proposals to the CNN every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

Both of the above algorithms(R-CNN & Fast R-CNN) uses selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore an object detection algorithm was proposed in **Faster-RCNN** that eliminates the selective search algorithm and lets the network learn the region proposals. Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

2.3.1 YOLO: You Only Look Once

YOLO [6] is a clever CNN for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression it then outputs recognized objects together with the bounding boxes. With YOLO,

a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

Steps

1. YOLO is based on the idea of segmenting an image into smaller images. The image is split into a square grid of dimensions $S * S$. The cell in which the center of an object resides, is the cell responsible for detecting that object.

2. Each cell will predict B bounding boxes and a confidence score for each box. Each of these bounding boxes is made up of 5 numbers: the x position, the y position, the width, the height, and the confidence.

3. The coordinates (x, y) represent the location of the center of the predicted bounding box, and the width and height are fractions relative to the entire image size. The confidence represents the Intersection Over Union (IOU) between the predicted bounding box and the actual bounding box, referred to as the ground truth box.

4. Each cell also predicts the class of the object. This class prediction is represented by a one-hot vector length C, the number of classes in the dataset. However, while each cell may predict any number of bounding boxes and confidence scores for those boxes, it only predicts one class.

5. Each prediction from a grid cell will be of shape $C+B*5$. Because there are $S*S$ grid cells in each image, the overall prediction of the model is a tensor of shape $S * S * (C + B * 5)$.

Architecture

The YOLO model is made up of three key components:

1. The Backbone

   It is the part of the network made up of convolutional layers to detect key features of an image and process them. The backbone is first trained on a classification dataset and

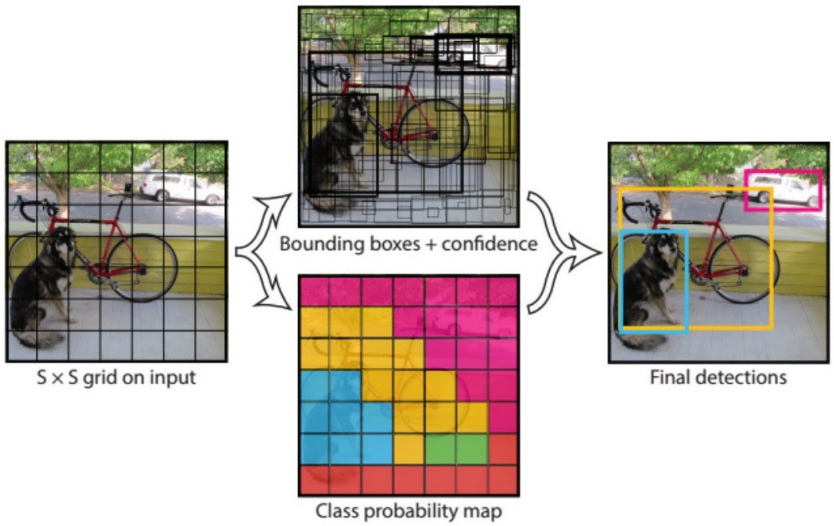


Figure 2.5: YOLO detection steps

typically trained at a lower resolution than the final detection model, as detection requires finer details than classification. YOLOv4 have CSP-Darknet-53 as the backbone of the network

2. The Neck

   It uses the features from the convolution layers in the backbone with fully connected layers to make predictions on probabilities and bounding box coordinates.

3. The Head

   It is the final output layer of the network which can be interchanged with other layers with the same input shape for transfer learning.

These three portions of the model work together to first extract key visual features from the image then classify and bound them. This happens for multiple scales. Each YOLO head corresponds to particular scale of detection. They are the combined on score values to produce final detection. Figure 2.6 shows the architecture of YOLOv4 model.

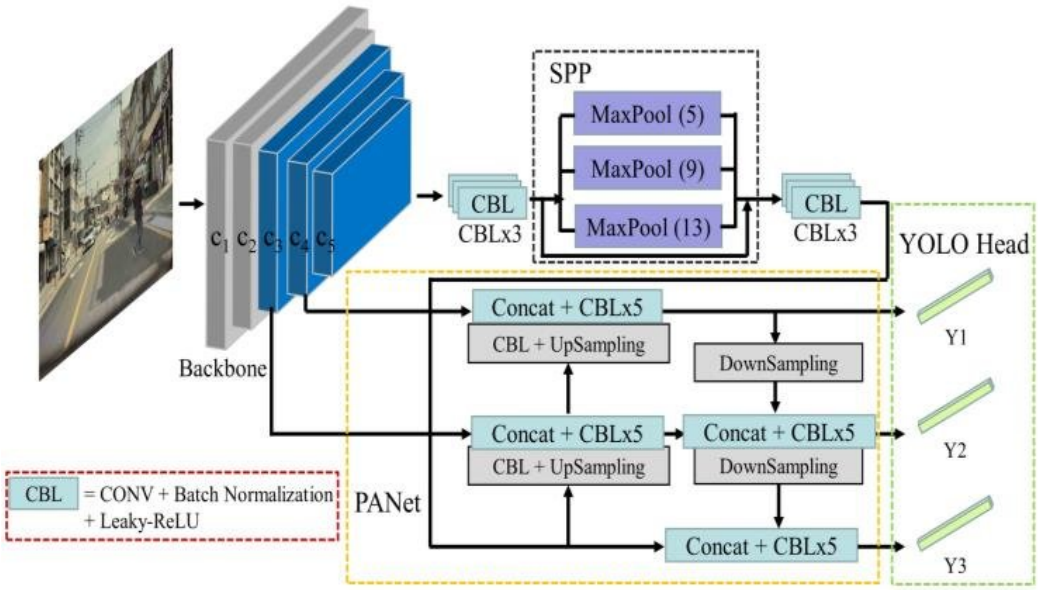


Figure 2.6: YOLOv4 architecture

YOLO variants

Varient	Release	FPS	mAP	Feature Extractor
YOLOv1 (448*448)	6 May 2016 Joseph Redmon	45	63.4	24 CNN 2 FC
YOLOv2 (416*416)	25 Dec 2016 Joseph Redmon	67	76.8	VGG-16
YOLOv3 (416*416)	8 April 2018 Joseph Redmon	35	55.3	Darknet-53
YOLOv4 (416*416)	23 April 2020 Alexey Brochkovskiy	38	62.8	CSPDarknet-53
YOLOv5	18 May 2020 Glenn Jocher	70.9	66.9	CSPDarknet-53
PP-YOLO (416*416)	3 August 2020 Xiang Long	72.9	62.8	ResNet50-vd-dcn

Table 2.1: YOLO variants comparison

2.3.2 DeepSORT: SORT with Deep Association

Simple Online and Realtime Tracking (SORT) [7] is a pragmatic approach to multiple object tracking with a focus on simple, effective algorithms. DeepSort [8] is an integration to

improve the performance of SORT. Due to this extension, we are able to track objects through longer periods of occlusions, effectively reducing the number of identity switches. Much of the computational complexity is spend in learning a deep association metric. During online application, we establish measurement-to-track associations using nearest neighbor queries in visual appearance space.

Traditional SORT algorithm simply uses the Kalman filter along with Hungarian algorithm for the tracking components. It achieves a speed of 260Hz. Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time frame. The Hungarian method is a combination optimization algorithm that solves the assignment problem in polynomial time and which anticipated later primal–dual methods. It gives a good tracking of object. But the id association fails where object osculation or sudden change in motion happens. It fails when the object is completely covered by another object. A deep association provides better re-identification even after skipping few frames. The feature vectors are matched using cosine distance similarity [9].

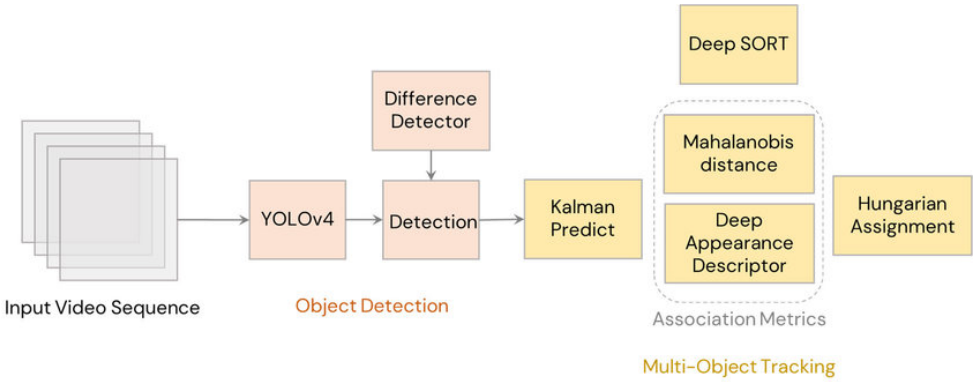


Figure 2.7: DeepSORT architecture

Figure 2.7 depicts architecture of DeepSORT. Frames of a series of time-series are passed into image detection model (Here YOLOv4). The model computes the bounding box for each detection. They are passed into a Detection Module to extract the feature vector, also called as 'encoders'. They are passed into Kalman filter for prediction. Cosine similarity is used

to associate ID. It is supported by Deep metric (Deep appearance Descriptor) and finally the next location is predicted in polynomial time using Hungarian Assignment technique. Several articles [10] [11] and GitHub repository [12] shows the implementation of Deepsort for Human re-identification.

2.3.3 Siamese Network

A Siamese neural network (also called a twin neural network) is an artificial neural network that uses the same weights while working on two different input vectors to compute comparable output vectors. The are often used for face matching, fingerprint matching, recognizing handwritten checks, and matching queries with indexed documents etc. It have a simple and unique architecture apart form other neural networks as shown in figure 2.8.

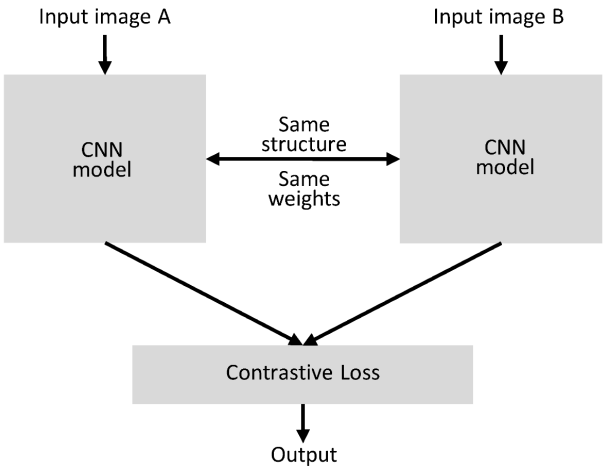


Figure 2.8: Siamese Network architecture

The network consists of two identical CNN models for feature extraction. The CNN model is trained separately to identify the images. The last classification layer of the separately trained CNN model is remove so that the feature vector is extracted. The feature vector from the two identical CNN model is passed into DNN layer of Siamese network that outputs if the images are similar or not. One of the image is called the anchor image and the other is the test images. Keras article [13] and GitHub repository [14] provides excellent reference materials.

Chapter 3

System Development

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

3.1 System Architecture

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus

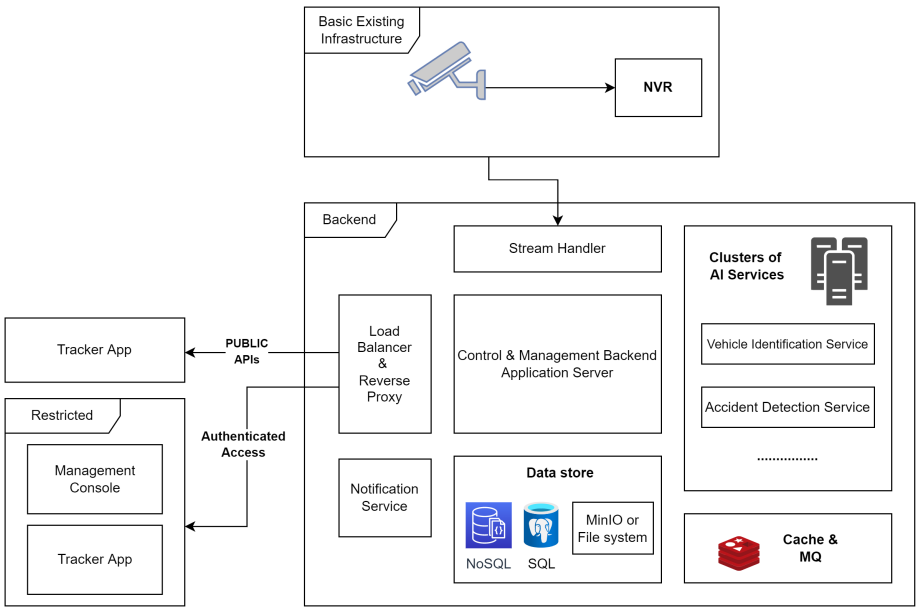


Figure 3.1: High level system architecture

sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra

metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3.1.1 Existing CCTV Infrastructure

Modern Infrasturcture of CCTV (Closed-Circuit Television) Video surveillance system pan out to a Network connected IP camera, which is streamed to an NVR (Network Video Recorder). Popular streaming protocols like RTSP (Realtime Streaming Protocol), RTSPS (RTSP over SSL), RTMP (Real-Time Messaging Protocol) are utlised in most IP cameras and NVR.

NVR stores the footage and is usually removed after specificed period based on the storage capacity of the NVR and number of cameras connected. On most NVR firmares, playback and downloading of recording is only provided by builtin web interface, or propriotary softwares bundled, which are known to be poorly implemented. Some NVR interface only works with full features on outdated Internet Explorer after installing plug-in.

Live streams of NVR and IP cameras can be accessed easily as RSTP stream using the network ip address, which can be played by any client in the same network.

3.1.2 VPN based Secure network access

CCTV Video surveillance systems utlise restricted closed network isolated from the internet, to reduce cyber attack vectors and unauthorised usage. Unsecured connection and system usage can

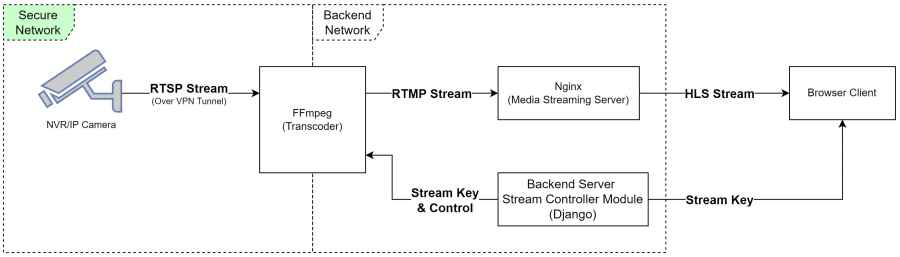


Figure 3.2: Camera Feed Live Streaming

compromise the security of network. Constructing a VPN (Virtual Private Network) split tunnel can enable secure communication interface to the network over unsecured channels. Wiregaurd, distributed in Linux Kernel, is utlised in our design to create VPN interface with very less overhead and very high performance.

3.1.3 Video Stream Ingestion

Video stream from NVR and IP camera are ingested to our system by Stream Handler and FrameGen module for proccessing using OpenCV. FrameGen module reads the live stream of cameras specified and save frames to storage backend with timestamp and signature as metadata. It ignores dead frames, blacklisted and idle frames and saves only required frames thereby reducing storage requirement.

Additionaly, FFmpeg is used for on-the-fly transcoding (transcoding while streaming) of live RSTP stream (which is a pull stream) from IP camera and NVR, to RTMP stream (which is a push stream) to NGINX Media Server. A stream key is attached to each stream which NGINX relays as HLS (HTTP Live Streaming) securely over SSL (Secure Socket Layer) to client browser of the authenticated user containing stream key.

Technology Stack

A hybrid microservice architecture is utilized to gain the scalability and decouple AI engine and services to serve web request efficiently, while ensuring quicker and simpler development and

deployment. Each independent services are created as a separate modular microservice which can be integrated based on requirement and infrastructure availability.

Development is primarily done in Python, while JavaScript is utilized for front-end scripting.

The list of frameworks and libraries used represent those which are widely utilized or have significant impact in the development.

Languages

- Python (Primary Language): Used in all server-side code.
- JavaScript: Used in front-end Scripting, Web-socket Browser Client, OpenPlayerJS

Frameworks

- Django: Back-end Server
- FastAPI: AI Engine API web framework and Web-socket Abstraction
- Keras & TensorFlow: AI inferencing
- Darknet & CUDA: Training Model

Other Tools

- Nginx: Used as web server and reverse proxy for handling request, serving static files, proxying request to application servers.
- Nginx RTMP Module: Used to enhance Nginx to be used as a Media Streaming Server
- PostgreSQL: As Relational Database
- WireGaurd: Creating secured encrypted VPN tunnel to NVR infrastructure network
- FFmpeg: For on-the-fly video transcoding to relay Live Camera RSTP feed to nginx as RTMP ingress feed

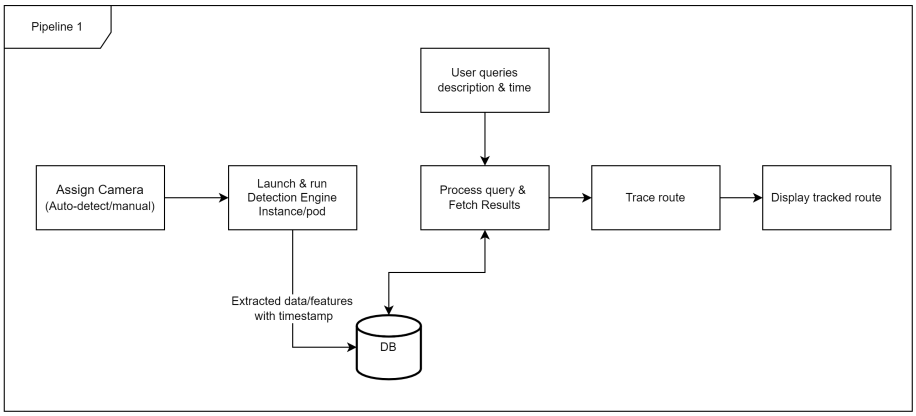
- OpenCV: Image Proccessing in FrameGen module and handling camera feed.
- WebSocket: Bidirectional server push communication with user
- OpenPlayerJS: Browser-based HLS video stream playback
- MinIO: File and object storage back-end to replicate Amazon Web Service S3.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

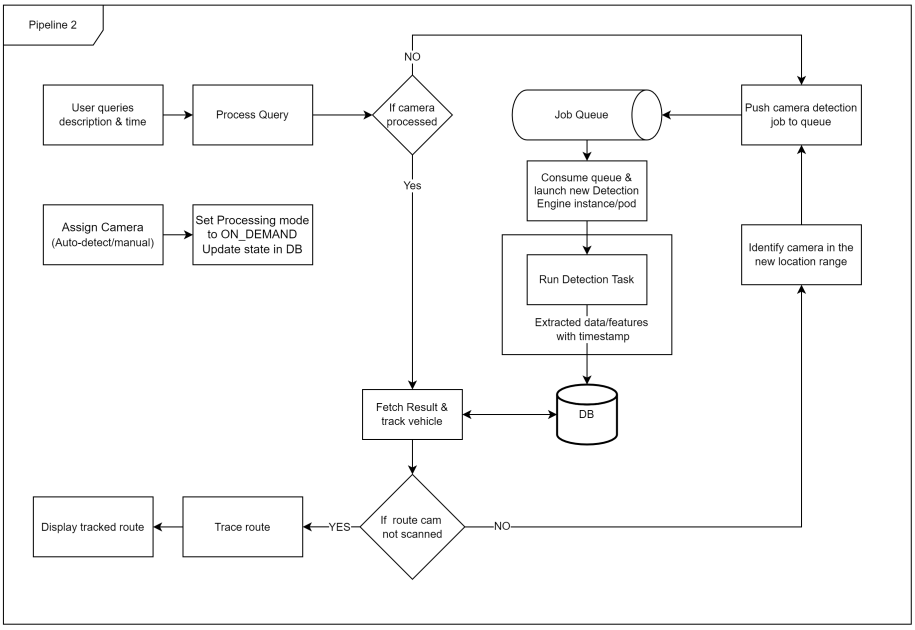
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3.2 Camera Network

The camera feed is obtained from a project deployed by Center for Information, Communication and Educational Technology (CICET), Government College of Engineering Kannur. Under the project entitled ”Third eye”, a virtual network was established covering an area of 350 KM^2, connecting 9 institutes. Using the existing cable network, various camera at installed allowing



(a) pipeline1



(b) pipeline2

Figure 3.3: Workflow pipeline

easy and secure storage and streaming. Currently the project houses holds about 190 cameras. Excess to these footage if obtained by secured virtual private junction.

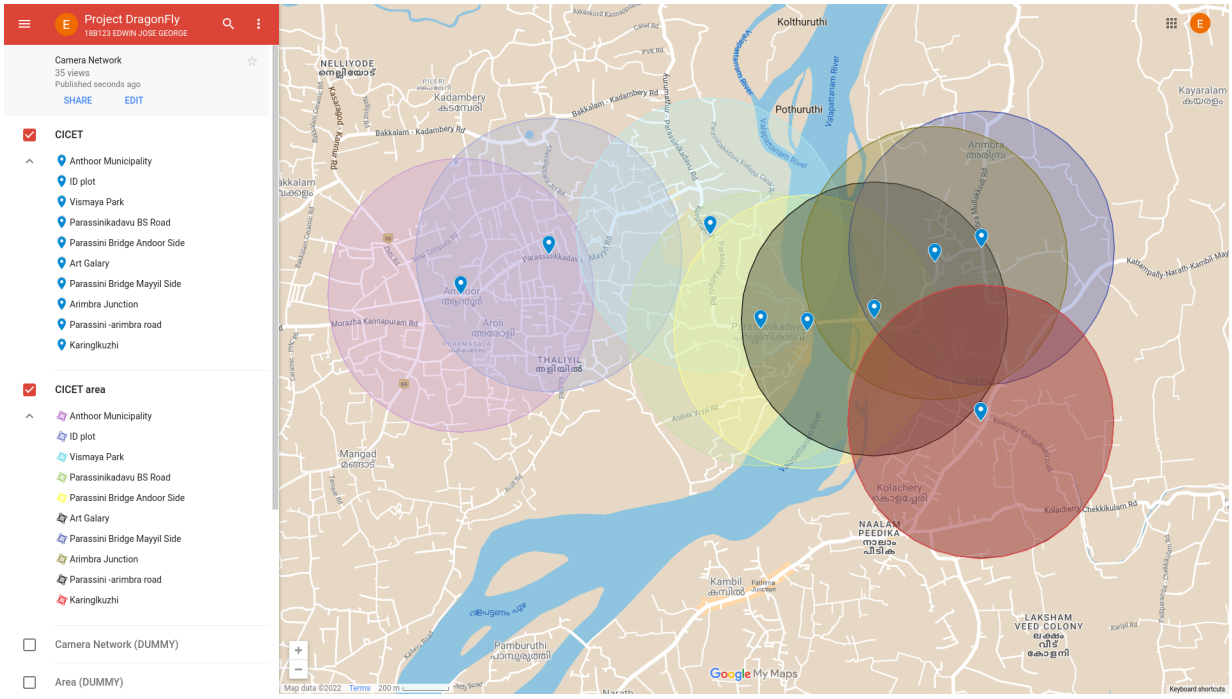


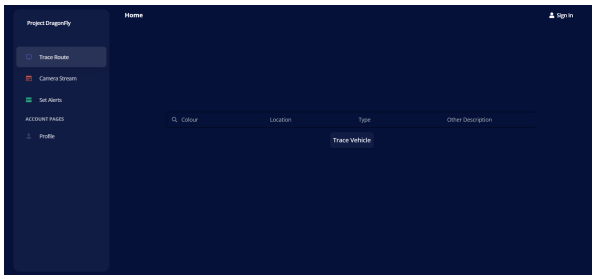
Figure 3.4: CICET camera network

Figure 3.5: Sample camera footage

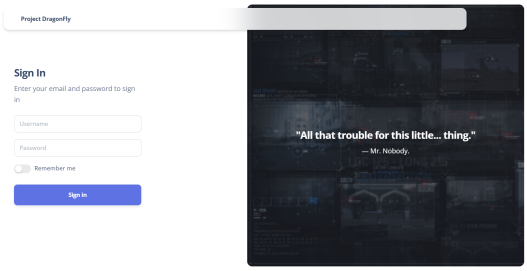
3.3 UI Design

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant

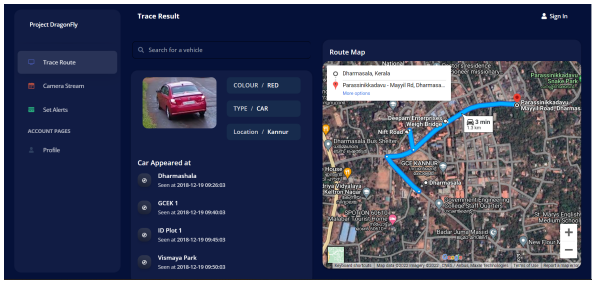
morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.



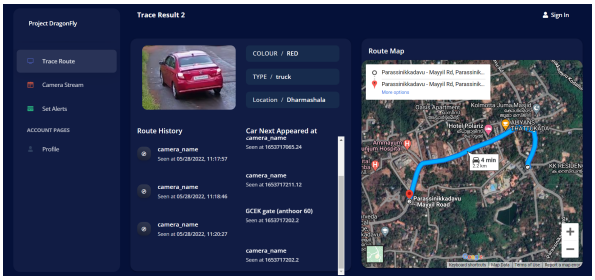
(a) Home page



(b) Login page



(c) Result



(d) Tracing

Figure 3.6: UI Design

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi,

congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

3.4 AI Model

YOLOv4

YOLOv4 model is used for the detection of vehicles. Transfer learning was conducted on the AI model that significantly reduced the training time required. Transfer learning is the process of using a pre-trained model and changing its parameter to learn new/fine tuned concept. Pre-trained weight was extracted from the official YOLOv4 repository [15] that used MS COCO dataset to train 80 classes. Parameter tuning is performed which are detailed in table 3.1.

Input size	$416 * 416$
Input Channels	3
Batch size	32
learning rate	0.0013
Yolo layer	$[256 * 256], [512 * 512], [1024 * 1024]$
Total Layers	162
Target Classes (9)	auto, bus, tempo traveler, tractor, truck, van, two wheeler, car, jcb

Table 3.1: YOLOv4 Parameter

The model was trained using Darknet at Google Colab. Later the environment was changed to the Workstation provided by the Dept. of Computer Science and Engineering, Govt. College of Engineering Kannur. A total of approx. 48 hours are spend in training, reaching 97.7% mean Average Precision (mAP) at 7300 iterations. Figure 3.7 summarizes training.

The dataset for training was collected at two different sources. Kaggle provided a wide variety of Indian Vehicle dataset that spanned across country. Footage from CICET camera network is also extracted. Each and every frame was labeled using LabelImg tool. Summary of the labeled dataset is depicted in the table 3.2.

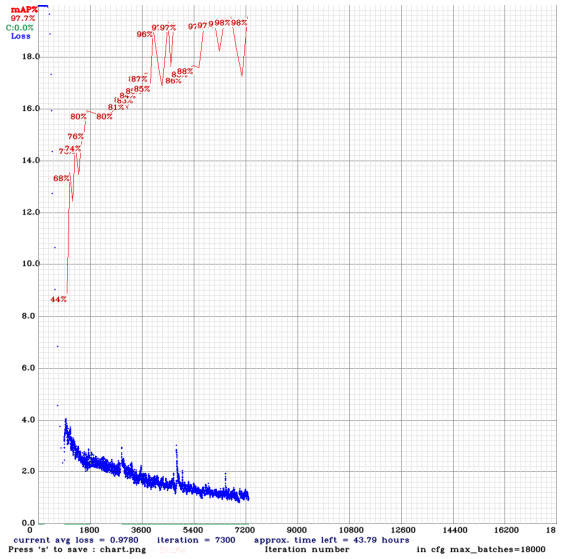


Figure 3.7: YOLOv4 training chart

Class	Kaggle	CICET	Total
Two wheeler	557	291	**848**
Truck	354	170	**424**
Auto	297	146	**443**
car	233	432	**665**
bus	220	112	**332**
tractor	133	0	**113**
van	101	131	**232**
JCB	1	0	**1**
Total Boxes	**1956**	**1282**	**3238**
Total Images	**733**	**826**	**1559**

Table 3.2: Dataset summary

DeepSORT

DeepSORT is utilized to assign unique id to each object in an continuous frame. The implementation is directly extracted from the article of AIGuysCode [11]. The given model tries to extract human features as the deep matrix. No modification are made as Kalman filter provides greater accuracy. However, learning the deep matrix of vehicles can further increase accuracy and aid in siamese network.

3.4.1 Image Dictionary

In-order to facilitate correct labeling of images, a concept called Image Dictionary is introduced. Image Dictionary is a hierarchical classification of images. Each type can have its sub-type. A visual representation is also provided for each type. It is build using D3 JS.

3.5 Algorithm

Algorithm 1 Process and store detections ∀ camera

INPUT : *video*, *camProp*

Figure 3.8: Image Dictionary

OUTPUT : $id, class, bbox, time, featureTensor \rightarrow Database$

$FPS \leftarrow video.FPS$
$time \leftarrow camProp.startTime$
$tracker \leftarrow$ TRACKER($cosineMetric$)
for all $frame \in video.frames$ **do**
    $names, bboxes, scores \leftarrow$ YOLOv4($frame$)
    $features \leftarrow$ ENCODER($frame, bboxes$) ▷ feature extraction
    $detections \leftarrow []$
    for $i \in$ LENGHT($names$) **do**
        $detections[i] \leftarrow (names[i], bboxes[i], scores[i], features[i])$
    end for
    TRACKER.PREDICT()
    TRACKER.UPDATE($detections$) ▷ DeepSORT assigns id
    for all $track \in tracker.tracks$ **do**
        if notTRACK.ISCONFIRMED() **then** ▷ Skip first few frame to ensure objects presence
            Continue
        else if $track.time_since_update \geq 1$ **then** ▷ Missing object in current frame
            Continue
        end if
        DATABASE.WRITE($track.id, track.class, track.tbwh, time, track.feature$)
    end for

```
    time ← time + 1/FPS
end for
```

Algorithm 2 Route building

INPUT : *vehicleClass*, *initialCamera*, *startTime*, *endTime*
OUTPUT : *route*

```
terminate ← FALSE
route ← []
vehicleList ← []
currentCam ← initialCamera
DB ← currentCam.DataBase                                   ▷ Find the vehicle
DB ← DB.FILTER(name = vehicleClass)
fDB ← DB.FILTER(time ≥ startTime)
fDB ← fDB.FILTER(time ≤ endTime)
for all vid ∈ UNIQUEID(fDB) do
    vDB ← DB.FILTER(id = vid)
    enterTime ← MIN(vDB.time)
    exitTime ← MAX(vDB.time)
    VEHICLELIST.APPEND(vid, enterTime, exitTime, currentCam)
end for
while terminate ≠ FALSE do                                 ▷ Search route iteratively
    if COUNT(vehicleList) == 0 then
        terminate ← TRUE
        Break
    else if COUNT(vehicleList) == 1 then
        seletedVehicle ← vehicleList[0]
    else
        selectedVehicle ← USERSELECT(vehicleList)
    end if
    ROUTE.APPEND(selectedVehicle)                          ▷ Add to route list
    addedCam ← selectedVehicle.camera
    nextCameras ← ADDEDCAM.NEIGHBOR(radius = 1KM)
    vehicleList ← []
    for currentCam ∈ nextCameras do
        startTime ← selectedVehicle.exitTime + MINTRAVELTIME(addedCam, currentCam)
        endTime ← selectedVehicle.exitTime + MAXTRAVELTIME(addedCam, currentCam)
        DB ← currentCam.DataBase                           ▷ Find the vehicles
        DB ← DB.FILTER(name = vehicleClass)
        fDB ← DB.FILTER(time ≥ startTime)
        fDB ← fDB.FILTER(time ≤ endTime)
        for all vid ∈ UNIQUEID(fDB) do
            vDB ← DB.FILTER(id = vid)
```

```
37:             enterTime ← MIN(vDB.time)
38:             exitTime ← MAX(vDB.time)
39:             VEHICLELIST.APPEND(vid, enterTime, exitTime, currentCam)
40:         end for
41:     end for
42: end while
43: PRINT(route)                                              ▷ The final result
```

Chapter 4

Results and Discussion

4.1 Camera Network

Camera Network provided a great asset in acquiring the relevant dataset. The system is aimed at real-time processing of multiple camera feed. However, minor steps backs are faced during the footage capture. The support for network is maintained by a Chinese video surveillance manufacturer, abbreviated as UNV. The software interface have lower support for live streaming of footage, which leads to complex method of acquiring frame. The camera occasionally provides with corrupted frames (see figure 4.1), leading to false detection. Camera footage is extracted at 25 FPS, $1920 * 1080$ px.

4.2 YOLOv4

YOLOv4 model is trained with 1559 images for 9 classes as illustrated in table 3.2. Model accuracy is easily calculated by Darknet and is shown in table 4.1 and 4.2. The model, in the deployment ran at slower speed of about 19.1 FPS. The decrease in speed is due to the unwanted processing of dead/corrupted frame. Speed can also be improved by reducing the yolo network size, compromising accuracy.

Figure 4.1: Corrupted Camera footage

Class label	True Positive	False Positive	Avg Precision
auto	424	38	98.25%
bus	327	42	99.57%
tempo traveler	76	7	97.12%
tractor	130	2	98.44%
truck	516	33	99.28%
van	227	22	98.14%
two wheeler	770	241	91.62%
car	644	91	96.99%
jcb	0	0	100.00%

Table 4.1: YOLOv4 accuracy matrix

Precision	87%
Recall	96%
F1-score	91%
True Positive	3114
False Positive	476
False Negative	134
Avg IoU	72.13%
Mean Avg precision	97.71%
Total Detection Time	1166 Seconds
Detection count	12349
Unique truth count	3257

Table 4.2: YOLOv4 evaluation

Figure 4.2: YOLOv4 vehicle predictions

4.3 DeepSORT

Ground Truth dataset was not available to quantize the result. However, via visual inception, it was found that the model was able to assign correct ID number to most of the vehicles. It was observed that such an increase in accuracy was observed due to the presence of Kalman filtering algorithm. The deep matrix association for humans was trigger only at the presence of humans. It lead to a mismatch where traditional Kalman filter tries to re-Id the bounding boxes, where as deepSORT tries to match human description leading to wrong answer. The anomaly can be strongly seen in public transport bus. Building of deep matrix for vehicle can improve the system.

4.4 Siamese Network

Proper Siamese Network was unable to construct, as the layers of yolov4 models was not able to selectively extract. The YOLOV4 model have tight coupling between layers, with multiple output and input vectors. The model is originally implement in Darknet. This provided hindrance during the extraction of layers using keras and python.

4.5 Other Limitations

- Current detection rate is about 19.1 FPS, which is not ideal to process multiple camera footage concurrently in real-time.
- The model was not trained to extract color class labels. Use of K-means gives poor realization of vehicle color. Background colors are also being extracted in K-means.
- User needs to visually inspect the similarity of target vehicle across the network, Siamese network is not completely realized.
- Limited user parameters for vehicle description. All parameter needs to be provided for search to begin.
- Heavy system resource utilization. About 5GB of GPU is being completely reserved and locked by YOLO model. Parsing larger network require higher RAM and CPU utilization. Larger volume of footage needs to be processed and stored.

Chapter 5

Conclusions and Future Scope

5.1 Conclusion

The project is developed using the existing technology and methods. This project extends the functionality of camera network offered by CICET. The footage are captured and processed by AI services. YOLOv4 model computes the bounding box along with the class confidence. DeepSORT uses the deep matrix to conduct multi-tracking in multiple objects. Upon the user query, the most relevant detections are provides, which the user can visually verify. This leads to the development of route parsed by the target vehicle. This project thus helps fast tracking of vehicles. The project can be considered as one of candidates for Open AI challenge proposed by NVIDIA.

5.2 Future Scope

Various levels of improvement can be done, specially in AI domain. Since False Positive do not have heavy impact on the problem domain, a fair trade-off can be applied between the speed and accuracy of the model. Heuristics can be applied after conducting the deeper analysis of the traffic pattern. Various development of PP-YOLO, a PyTorch implementation of YOLO model, is gaining popularity. A new framework called PaddlePaddle [16] is being developed

aiming distributed processing, which can significantly increase the processing speed. Various methods are being proposed in PaddlePaddle that significantly reduces the complexity of several algorithms, making real-time application more realizable. Below are some of the pointer that can further enhance the functionality of the project:

- Usage of higher resolution camera can better extract the unique feature of vehicles. Cameras can be placed under a bright light source enabling better detection at night.
- Integration of license plate detection and Motor Vehicle Department can be used to reverse map the vehicle description, in search of forged vehicle registration.
- Train the model in varying weather condition such as sunny, cloudy, rainy etc. Target classes can be increased to detect sub-category of vehicles.
- Tighter integration of YOLO model and DeepSORT can reduce redundant extraction and processing of feature vectors.
- Current route generation requires visual inspection, which can be replaced by successful implementation of Siamese Network.
- Mapping between camera needs to be improved for better route estimation.
- Use of elastic search at user interface for interaction and extraction of feature description.
- Secured IP streaming of camera feed using DRM protection.
- Use of PP-YOLO with PaddlePaddle framework can boost the overall performance of the distributed system.

References

[1] B. K, "Object detection algorithms and libraries - neptune.ai," 2022. [Online]. Available: https://neptune.ai/blog/object-detection-algorithms-and-libraries

[2] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 8789–8798. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2019arXiv190309254T

[3] M. A. Manzoor, Y. Morgan, and A. Bais, "Real-time vehicle make and model recognition system," *Machine Learning and Knowledge Extraction*, vol. 1, no. 2, pp. 611–629, 2019. [Online]. Available: https://www.mdpi.com/2504-4990/1/2/36

[4] J. Cai, J. Deng, M. U. Aftab, M. S. Khokhar, R. Kumar *et al.*, "Efficient and deep vehicle re-identification using multi-level feature extraction," *Applied Sciences*, vol. 9, no. 7, p. 1291, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/7/1291

[5] J. Deng, Y. Hao, M. S. Khokhar, R. Kumar, J. Cai, J. Kumar, M. U. Aftab *et al.*, "Trends in vehicle re-identification past, present, and future: A comprehensive review," *Mathematics*, vol. 9, no. 24, p. 3162, 2021. [Online]. Available: https://www.mdpi.com/2227-7390/9/24/3162

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015. [Online]. Available: https://arxiv.org/abs/1506.02640

[7] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2016. [Online]. Available: https://doi.org/10.1109%2Ficip.2016.7533003

[8] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649. [Online]. Available: https://arxiv.org/abs/1703.07402

[9] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 748–756.

[10] S. R Maiya, "Deepsort: Deep learning to track custom objects in a video," 2022. [Online]. Available: https://nanonets.com/blog/object-tracking-deepsort/#deep-sort

[11] theAIGuysCode, "yolov4-deepsort," https://github.com/theAIGuysCode/yolov4-deepsort, 2021.

[12] B. Chen, "Object-detection-and-tracking," https://github.com/yehengchen/Object-Detection-and-Tracking, 2021.

[13] K. Team, "Keras documentation: Image similarity estimation using a siamese network with a triplet loss," 2022. [Online]. Available: https://keras.io/examples/vision/siamese_network/

[14] A. Dutt, "Siamese-networks-tutorial," https://github.com/AdityaDutt/Siamese-Networks-Tutorial, 2021.

[15] A. Brochkovskiy, "Darknet," https://github.com/AlexeyAB/darknet, 2022.

[16] P. org, "Paddledetection," https://github.com/PaddlePaddle/PaddleDetection, 2022.