Mini-Project Report On

AUTOMATED OVERTAKING DETECTION SYSTEM FOR BRIDGES

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

HARIKRISHNAN R (U2004039)

Under the guidance of
Mr. HARIKRISHNAN M

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"AUTOMATED OVER-TAKING DETECTION SYSTEM FOR BRIDGES"*** *is a bonafide work done by* ***Mr.HARIKRISHNAN R (U2004039)*** *, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.
Head of Department
Dept. of CSE
RSET

Mr. Uday Babu P.
Mini-Project Coordinator
Asst. Professor
Dept. of CSE
RSET

Mr. HARIKRISHNAN M
Mini-Project Guide
Asst. Professor
Dept. of CSE
RSET

ACKNOWLEDGEMENTS

ABSTRACT

In this project I have developed an automatic overtaking detection system that helps to identify people who overtake vehicles in bridges, accurately recording the violators' details to the database. The officials upload a camera footage, the program then uses machine learning and object detection to identify the violators, take the image, understand and extract the part of the image that shows the number plate of cars. I then use optical character recognition (OCR) to extract the characters from the region of interest and add it to the database. The admin can then view the database and send the fee receipt to the vehicle owner's Email.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

1.2 Existing System

Kerala government has launched the 'Safe Kerala' project starting a few days back that will use artificial intelligence (AI) cameras to identify traffic violations and impose fines in the state.Initially the AI cameras will detect violations like driving a two-wheeler without wearing a helmet, traveling without wearing a seat belt, more than two people traveling on a two-wheeler, using mobile phones while driving, and traffic signal violation.Fully Automated Traffic Enforcement System" will inform the vehicle owner of the traffic violation via courier, e-mail and mobile phone within 24 hours of the alleged offence.

1.3 Problem Statement

My project helps in identifying and monitoring vehicles that are illegally overtaking other vehicles in a bridge.

This involves detecting when a vehicle is attempting to overtake another vehicle in the same lane, and then taking appropriate action to prevent the violation from occurring.

It is threat for pedestrians and other drivers as it covers visibility leading to collision.

1.4 Objectives

To detect traffic violation on bridges and to improve road safety.

1.5   Scope

Currently the AI cameras and traffic cameras are meant to capture violations happening on roads but we do not have a system to capture overtaking on bridges.

So I am implementing this project to execute that.

Chapter 2

Literature Review

2.1   Object Detection using YOLO Algorithm

Object detection is a computer vision task that involves identifying and locating objects in images or videos. It is an important part of many applications, such as surveillance, self-driving cars, or robotics. Object detection algorithms can be divided into two main categories: single-shot detectors and two-stage detectors.

One of the earliest successful attempts to address the object detection problem using deep learning was the R-CNN (Regions with CNN features) model, developed by Ross Girshick and his team at Microsoft Research in 2014. This model used a combination of region proposal algorithms and convolutional neural networks (CNNs) to detect and localize objects in images.Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network.

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.Several new versions of the same model have been proposed since the initial release of YOLO in 2015

YOLO divides an input image into an S × S grid. If the center of an object falls into

a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

2.2 Image Modification using OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly for real-time computer vision.Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). The library is cross-platform and licensed as free and open-source software under Apache License 2. Starting in 2011, OpenCV features GPU acceleration for real-time operations. Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were describedas:

Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel. Disseminate vision knowledge by

providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable. Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself. The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

In May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.

In July 2020, OpenCV announced and began a Kickstarter campaign for the OpenCV AI Kit, a series of hardware modules and additions to OpenCV OpenCV is written in the programming language C++, as is its primary interface, but it still retains a less comprehensive though extensive older C interface. All newer developments and algorithms appear in the C++ interface. There are language bindings in Python, Java, and MATLAB/Octave. The application programming interface (API) for these interfaces can be found in the online documentation. Wrapper libraries in several languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions were released as OpenCV.js, to be used for web platforms.

2.3 Optical Character Recoginition using EasyOCR

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene photo (for

example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast).Pre-processing OCR software often pre-processes images to improve the chances of successful recognition. Techniques include:[15]

De-skewing – if the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical. Despeckling – removal of positive and negative spots, smoothing edges Binarization – conversion of an image from color or greyscale to black-and-white (called a binary image because there are two colors). The task is performed as a simple way of separating the text (or any other desired image component) from the background. The task of binarization is necessary since most commercial recognition algorithms work only on binary images, as it is simpler to do so. In addition, the effectiveness of binarization influences to a significant extent the quality of character recognition, and careful decisions are made in the choice of the binarization employed for a given input image type; since the quality of the method used to obtain the binary result depends on the type of image (scanned document, scene text image, degraded historical document, etc.). Line removal – Cleaning up non-glyph boxes and lines Layout analysis or zoning – Identification of columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables. Line and word detection – Establishment of a baseline for word and character shapes, separating words as necessary. Script recognition – In multilingual documents, the script may change at the level of the words and hence, identification of the script is necessary, before the right OCR can be invoked to handle the specific script. Character isolation or segmentation – For per-character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connected. Normalization of aspect ratio and scale Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas. For proportional fonts, more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character.

Text recognition There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.

Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as pattern matching, pattern recognition, or image correlation. This relies on the input glyph being correctly isolated from the rest of the image, and the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique early physical photocell-based OCR implemented, rather directly. Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. The extraction features reduces the dimensionality of the representation and makes the recognition process computationally efficient. These features are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and most modern OCR software.[24] Nearest neighbour classifiers such as the k-nearest neighbors algorithm are used to compare image features with stored glyph features and choose the nearest match. Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as adaptive recognition and uses the letter shapes recognized with high confidence on the first pass to better recognize the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded).

As of December 2016, modern OCR software includes Google Docs OCR, ABBYY FineReader, Rossum's AI OCR software, and Transym.[26][needs update] Others like OCRopus and Tesseract use neural networks which are trained to recognize whole lines of text instead of focusing on single characters.

A technique known as iterative OCR automatically crops a document into sections based on page layout. OCR is performed on the sections individually using variable character confidence level thresholds to maximize page-level OCR accuracy. A patent from the United States Patent Office has been issued for this method.

The OCR result can be stored in the standardized ALTO format, a dedicated XML schema maintained by the United States Library of Congress. Other common formats include hOCR and PAGE XML.

For a list of optical character recognition software, see Comparison of optical character recognition software.

Post-processing OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document.[15] This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

Near-neighbor analysis can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together.[28] For example, "Washington, D.C." is generally far more common in English than "Washington DOC".

Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy.

The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printed data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed online, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of accuracy for most fonts are now common, and with support for a variety of image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

2.4 Tensorflow : for training

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript.

In Jan 2019, Google announced TensorFlow 2.0.[19] It became officially available in Sep 2019.

In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, including Python,

JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

2.5   Pytorch

PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

A number of pieces of deep learning software are built on top of PyTorch, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, and Catalyst.

PyTorch provides two high-level features:

Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU) Deep neural networks built on a tape-based automatic differentiation system PyTorch defines a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable NVIDIA GPU. PyTorch has also been developing support for other GPU platforms, for example, AMD's ROCm and Apple's Metal Framework.

PyTorch supports various sub-types of Tensors.

Note that the term "tensor" here does not carry the same meaning as tensor in mathematics or physics. The meaning of the word in machine learning is only tangentially related to its original meaning as a certain kind of object in linear algebra.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system of user which is a smart phone is expected to have the following features:

- Windows should be version 10 or above

- Requirement of Internet connection for login and addition to database.

- A storage space of approximate 3 GB for the app.

- A minimum Ram size of 12GB is required in the device.

- Graphics Card with GPU: NVIDIA Ge Force GTX

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

This project can be accessed only by the Motor Vehicle Department Officials because the data from the camera should be secured and should not be accessed by other people.

3.2.2 Operational Feasibility

The operations are built in a simple and easy to use manner for the officials of the Motor Vehicle Department. They can get the number plate data from the app and issue fine receipts. Installation of the app will be done by the software developers.

3.3 Hardware Requirements

The following are the system requirements to develop the AODS App.

- Processor: Intel Core i5

- Hard Disk: Minimum 100GB

- RAM: Minimum 12GB

3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows 10 or above IDE: Pycharm 1.2 Python 3.10

3.4.1 Pycharm for python app development

PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django. PyCharm is developed by the Czech company JetBrains.

It is cross-platform, working on Microsoft Windows, macOS and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License.PyCharm Community Edition is less extensive than the Professional Edition.The beta version of the product was released in July 2010, with the 1.0 arriving 3 months later. Version 2.0 was released on 13 December 2011, version 3.0 was released on 24 September 2013, and version 4.0 was released on November 19, 2014.

PyCharm became Open Source on 22 October 2013. The Open Source variant is released under the name Community Edition – while the commercial variant, Professional Edition, contains closed-source modules.

3.4.2 PyTorch

PyTorch is an open-source machine learning library for Python that allows you to easily build, train, and deploy deep learning models. It is designed to be flexible, efficient, and easy to use, and it offers a wide range of features and capabilities that can be used for a variety of tasks, including computer vision, natural language processing, and scientific computing.

PyTorch provides a variety of tools and features that make it easy to build, train, and deploy deep learning models, including:

A powerful tensor library that allows you to perform operations on multi-dimensional arrays. A flexible neural network library that allows you to define and train complex models Utilities for loading and preprocessing data. Integration with popular optimization algorithms and loss functions. You can use PyTorch to build and train a wide range of machine learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory (LSTM) networks. PyTorch is widely used in research and industry, and it is a popular choice for building deep learning models.

Chapter 4

Methodology

4.1 Proposed Method

- Develop an application that can detect vehicles that overtake on bridges, detect their number plates and store it with the image of the violator.

- I build a pre trained model for image enhancement of the captured image in order to get quality images from which we get number plate details accurately.

- User gives a video footage as input and the application detects the violators.

- It can also tell how much time a particular vehicle has done traffic violation.

4.1.1 Vehicle Detection

Detecting vehicles in a video stream is an object detection problem. An object detection problem can be approached as either a classification problem or a regression problem. In the classification approach, the image are divided into small patches, each of which will be run through a classifier to determine whether there are objects in the patch. The bounding boxes will be assigned to patches with positive classification results. In the regression approach, the whole image will be run through a convolutional neural network directly to generate one or more bounding boxes for objects in the images. The YOLO approach of the object detection is consists of two parts: the neural network part that predicts a vector from an image, and the postprocessing part that interpolates the vector as boxes coordinates and class probabilities.

For the neural network, the tiny YOLO v1 is consist of 9 convolution layers and 3 full connected layers. Each convolution layer consists of convolution, leaky relu and max pooling operations. The first 9 convolution layers can be understood as the feature extractor, whereas the last three fully-connected layers can be understood as the "regression

head" that predicts the bounding boxes. There are a total of 45,089,374 parameters in the model.

Deep Convolutional Networks

In all our experiments we train the CNN using Stochastic Gradient Descent (SGD) with standard backprop and AdaGrad. In most experiments we start with a learning rate of 0.05 which we lower to finalize the model. The models are initialized from random, similar to, and trained on a CPU cluster for 1,000 to 2,000 hours. The decrease in the loss (and increase in accuracy) slows down drastically

We used two types of architectures and explore their trade-offs in more detail in the experimental section. Their practical differences lie in the difference of parameters and Floatings-Point Operations Per Second (FLOPS). The best model may be different depending on the application. E.g. a model running in a datacenter can have many parameters and require a large number of FLOPS, whereas a model running on a mobile phone needs to have few parameters, so that it can fit into memory. All our models use rectified linear units as the non-linear activation function. It has a total of 140 million parameters and requires around 1.6 billion FLOPS per image. The second category use is based on GoogLeNet style Inception models . These models have $20\times$ fewer parameters (around 6.6M-7.5M) and up to $5\times$ fewer FLOPS (between 500M-1.6B). Some of these models are dramatically reduced in size (both depth and number of filters), so that they can be run on a mobile phone. One, NNS1, has 26M parameters and only requires 220M FLOPS per image. The other, NNS2, has 4.3M parameters and 20M FLOPS. Table 2 describes NN2 our largest network in detail. NN3 is identical in architecture but has a reduced input size of 160x160. NN4 has an input size of only 96x96, thereby drastically reducing the CPU requirements (285M FLOPS vs 1.6B for NN2). In addition to the reduced input size it does not use 5x5 convolutions in the higher layers as the receptive field is already too small by then. Generally we found that the 5x5 convolutions can be removed throughout with only a minor drop in accuracy. Figure 4.3 compares all our models.

Chapter 5

System Design

5.1 Architecture Diagram
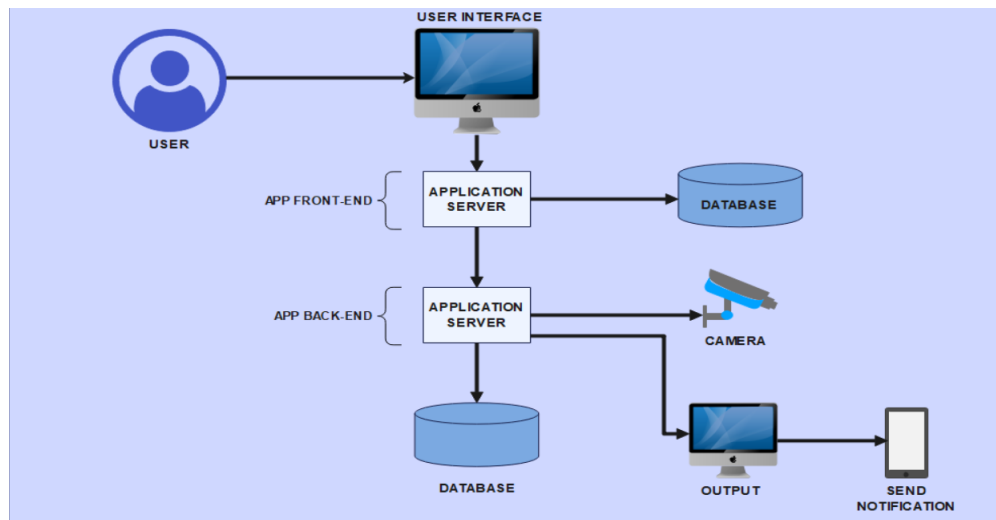
Figure 5.1: Architecture diagram

5.2 Use case diagram
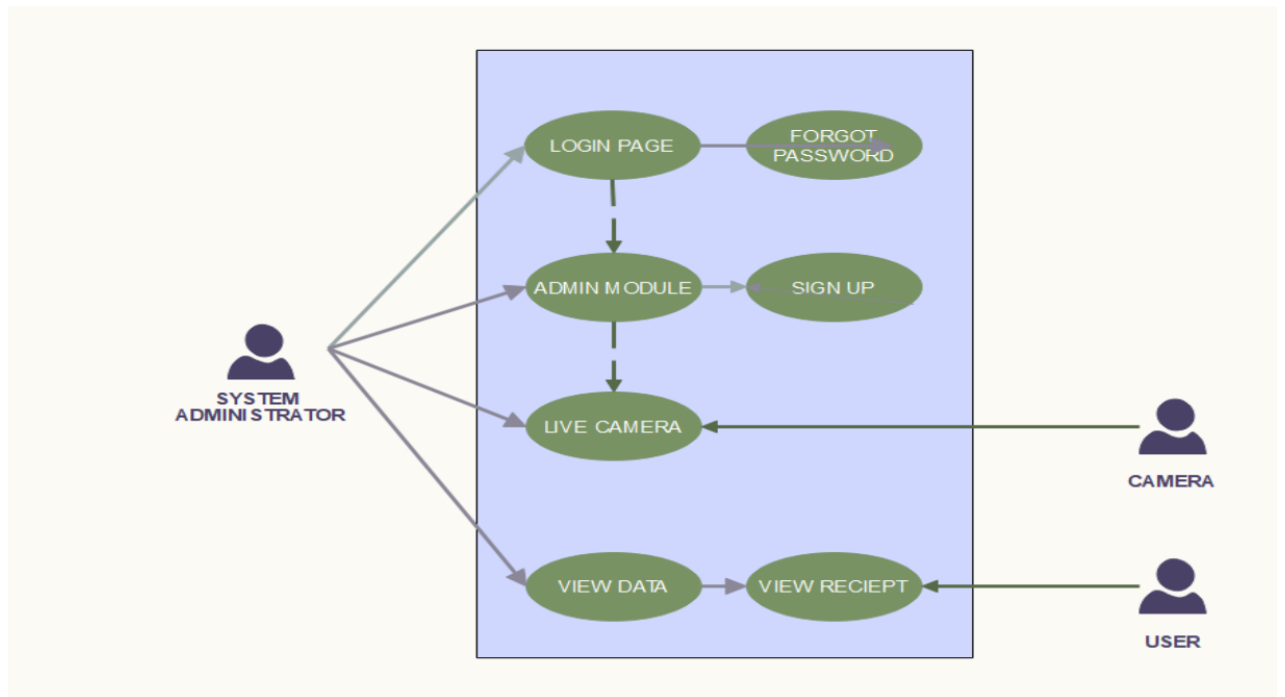
Figure 5.2: Use case diagram

Chapter 6

System Implementation

6.1   Login page

The app has a Login page which has Username, Password, Hide Password, Forgot Password and Login button fields. The details of the Username and Password are stored into a Database and will be taken when login page is used. The hide password button will toggle the password by showing it and hiding it. If the user forgets the password they should click the forgot password and an OTP will be sent to the registered Email ID and they can Reset the Password. The new password will be modified in the Database

6.2   Home Page

After Login the app will take the user to the Home Page. At the home page we have buttons for Live camera footage which will give us the footage of the traffic. There is a new user registration option for the officials to add a new user. After inputting Name, Email, Mobile number and Password the new user will be added to the database. It has the button Violators to show us all the Violators that are stored in the database. It will show the unique ID given to the vehicle, the register number and the image of the violator. The button No of violators we can see how many time a specific vehicle has done a traffic violation. The app can also switch Dark mode and Light mode. It can also resize the application by changing the scaling of the application

6.3   Vehicle detection

This project coverts the video into frames and will identify the vehicle that is overtaking on a bridge. The YOLO algorithm that identifies the violators uniquely and it will take the car uniquely and saves it with the date, time into a folder.

6.4   Number Plate detection and extraction

We take the image that was saved into the folder and we import it into the model that we have trained in order to find the number plate coordinates mask the remaining portion in the image and crop the number plate. It is the uploaded to another trained model for image enhancement for removal of noise from the image. The image then uses Easy-OCR to extract the number from the image. Since the data set is from a Thailand traffic footage we extracted the number with Thai language letters in it. It is then stored into the database with the vehicle ID.

Chapter 7

Testing

7.1 Testing

Testing is of utmost importance for the automated overtaking detecting sytem in bridges as it ensures the reliability and functionality provided by the application. Although testing played a significant role in identifying and addressing various issues, certain aspects could not be fully resolved within the tight 4-month development deadline. Despite these limitations, testing allowed for the identification of critical bugs and improvements, contributing to a more correct and unambiguous outcome.

We trained a dataset and ran 25 apoc to detect the numberplate from each vehicle which is about 400 to 450 pictures were tested and checked. Each time a vehicle crosses the white line (the line of references) that violates the traffic rule a png file of the violaters car is taken from it the number plates png file is also derived, it is then enhanced and stored .

7.2 Unit Testing

Individual modules or components of the platform were tested in isolation to confirm their correctness and accuracy. Writing test cases for each unit was used to carry out this testing ie image capturing, numberplate detection etc.

7.3 Integration Testing

After that to evaluate how well the platform's various components are integrated, integration testing was done and was tested.It included all the features that is no of violates,new login, violaters list , live cam.

7.4 System Testing

In order to verify that the system was in conformity with the project's requirements and goals, system testing required evaluating the system as a whole.

7.5 User Interface Testing

User interface (UI) testing was done to verify the visual components, design, and usability of the platform. It was done to see that all the functions in the gui are met .
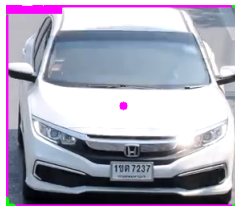
Figure 7.1: A violated car

Fig. 7.1. shows the image case when a car has overtaken or violated the traffic rule

Figure 7.2: Number plate of the car
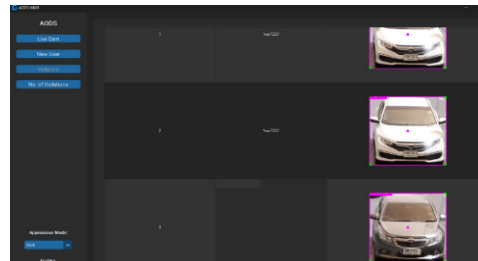
Figure 7.3: Violaters table (Shows how the car and registration number is stored in the Database)

From Fig. 7.2. and Fig. 7.3. We observe how a car's image is snapped and stored in the database along with the registration number.

Chapter 8

Results
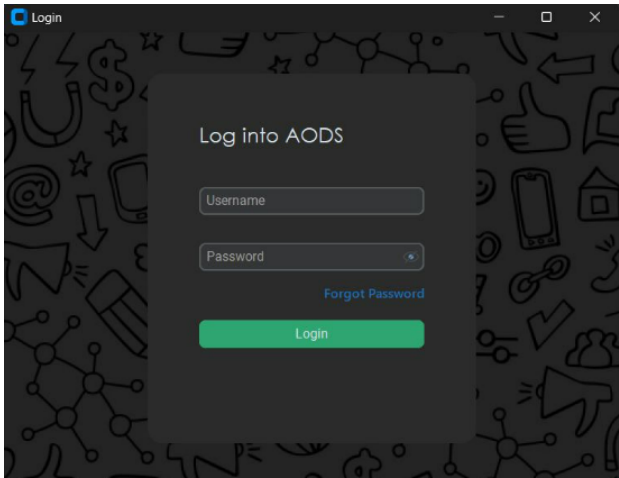
Figure 8.1: Login Page
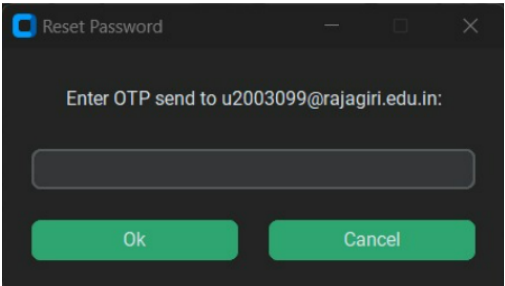
Figure 8.2: OTP Window
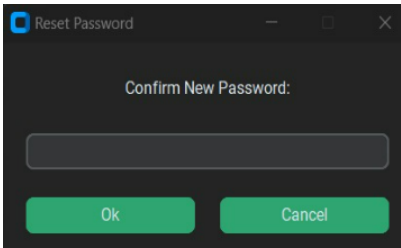
Figure 8.3: Password confirmation
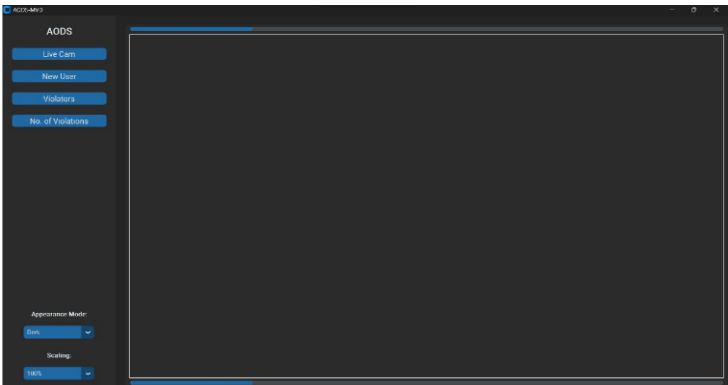
Figure 8.4: Home Page
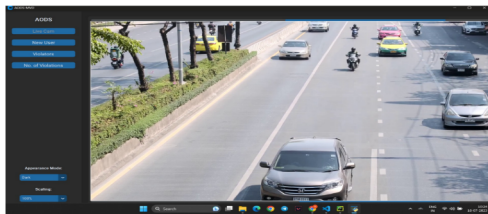
Figure 8.5: Live camera footage
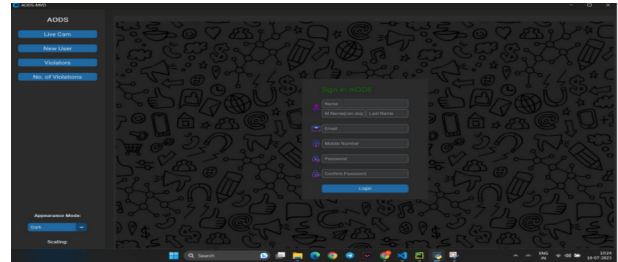
Figure 8.6: New user registration
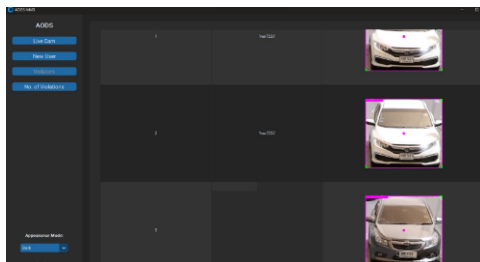
Figure 8.7: Violators List
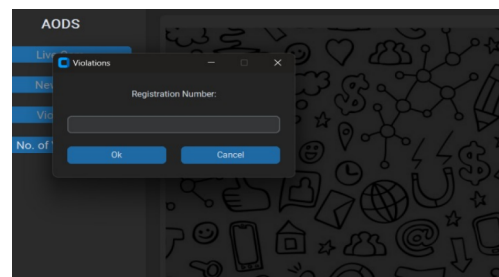
Figure 8.8: No of Violations

Chapter 9

Risks and Challenges

9.1 High Quality Camera

Selecting a high-quality camera for vehicle detection as a challenge that requires consideration of various factors to ensure optimal performance and accuracy. Here are some key points to consider when choosing the camera: Resolution,Frame rate,Low light performance,Dynamic range,Field of view,Weather resistance,Image processing capabilities,Connectivity,Mounting options, Compatibility with detection software and Reliability and durability.

9.2 Accurate Output

Achieving accurate output in vehicle detection is indeed a challenging task, and it requires careful consideration of various factors to improve the system's performance. Some key challenges and potential solutions to achieving accurate output: Post-processing,High-quality training data,Algorithm selection etc

9.3 High Quality GPU for Efficient Performance

The need for a high-end GPU can indeed present challenges in vehicle detection and other computationally-intensive tasks. Some of the challenges associated with this requirement: Cost, Hardware availability, Power consumption, Cooling requirements, Portability and Compatibility and integration

9.4 Data Scarcity

Data scarcity is a significant challenge in many machine learning tasks, including vehicle detection. When there is a limited amount of data available for training and testing, it can negatively impact the accuracy and generalization capabilities of the detection system.

Chapter 10

Conclusion

Detection and tracking of vehicles has been recognised and the violators has been detected.The register number recognition has to be recognised and stored to the database.This will help us in reducing accidents occurring in the bridges.

After addition of more new features we can also use this to detect the violations that occur during night. With a more improved data-set we can get more accurate results which can be useful for the Motor Vehicle Department to fine the violators. We can modify the code to find out number of vehicles passing through an area which can be helpful in controlling and analysing traffic in that area

References

[1] A. Kashyap, B. Suresh, A. Patil, S. Sharma and A. Jaiswal, "Automatic Number Plate Recognition," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2018, pp. 838-843, doi: 10.1109/ICACCCN.2018.8748287.

[2] Q. Huang, Z. Cai and T. Lan, "A Single Neural Network for Mixed Style License Plate Detection and Recognition," in IEEE Access, vol. 9, pp. 21777-21785, 2021, doi: 10.1109/ACCESS.2021.3055243.

[3] D. Padilla Carrasco, H. A. Rashwan, M. Á. García and D. Puig, "T-YOLO: Tiny Vehicle Detection Based on YOLO and Multi-Scale Convolutional Neural Networks," in IEEE Access, vol. 11, pp. 22430-22440, 2023, doi: 10.1109/ACCESS.2021.3137638.

[4] Y. Le Montagner, E. D. Angelini and J. -C. Olivo-Marin, "An Unbiased Risk Estimator for Image Denoising in the Presence of Mixed Poisson–Gaussian Noise," in IEEE Transactions on Image Processing, vol. 23, no. 3, pp. 1255-1268, March 2014, doi: 10.1109/TIP.2014.2300821.

[5] B. Xu, D. Zhou and W. Li, "Image Enhancement Algorithm Based on GAN Neural Network," in IEEE Access, vol. 10, pp. 36766-36777, 2022, doi: 10.1109/ACCESS.2022.3163241.

Appendix A: Base Paper

RESEARCH ARTICLE

Fast-Yolo-Rec: Incorporating Yolo-Base Detection and Recurrent-Base Prediction Networks for Fast Vehicle Detection in Consecutive Images

NAFISEH ZAREI[1], PAYMAN MOALLEM[ID][1], AND MOHAMMADREZA SHAMS[2]

[1]Department of Electrical Engineering, Faculty of Engineering, University of Isfahan, Isfahan 81746-73441, Iran
[2]Department of Computer Engineering, Shahreza Campus, University of Isfahan, Isfahan 81746-73441, Iran

Corresponding author: Payman Moallem (p_moallem@eng.ui.ac.ir)

ABSTRACT Despite significant advances and innovations in deep network-based vehicle detection methods, finding a balance between detector accuracy and speed remains a significant challenge. This study aims to present an algorithm that can manage the speed and accuracy of the detector in real-time vehicle detection while increasing detector speed with accuracy comparable to high-speed detectors. To this end, the Fast-Yolo-Rec algorithm is proposed. The proposed method includes a new Yolo-based detection network and LSTM-based position prediction networks. The proposed semantic attention mechanism in the spatial semantic attention module (SSAM) significantly impacts accuracy and speed on par with the most recent fast detectors. Recurrent position prediction networks, on the other hand, improve the detection speed by estimating the current vehicle position using vehicle position history. The vehicle trajectories are classified, and the LSTM network for the specified trajectory predicts the vehicle positions. The Fast-Yolo-Rec algorithm not only determines the position of the vehicle faster than high-speed detectors but also allows for the speed control of the detection network with acceptable accuracy. The evaluation results on a large Highway dataset show that the proposed scheme outperforms the baseline methods.

INDEX TERMS Yolo-based detection network, attention mechanism, recurrent prediction network.

I. INTRODUCTION

The detection and classification of vehicles in intelligent transportation systems play a crucial role in urban traffic management, reducing traffic violations, measuring vehicle speeds, and enabling more detailed violation evaluation [1], [2], [3], [4].

Convolutional neural networks (CNNs) are among the most successful methods for promoting object detection. They are excellent at learning image features and can perform various tasks related to classification and bounding box regression [5]. CNN-based methods are divided into one-stage and two-stage detection networks. One-stage detectors have high inference speeds. On the other hand, two-stage detectors have significant localization accuracy and low speed. These networks have revolutionized the detection

The associate editor coordinating the review of this manuscript and approving it for publication was Angel F. García-Fernández[ID].

of objects; however, despite recent advances, the tradeoff between the speed and accuracy of such networks remains a major challenge [6]. Two-stage detection networks, such as R-CNN [7], Fast R-CNN [8], and Faster R-CNN [9], focus on areas of the image that are more likely to contain the target rather than the entire image. Each of these areas is processed separately by the network. Due to a large number of selected regions, in the range of 1000 to 2000, the network can process the image thousands of times. While single-stage detectors, such as Yolo [10], process the image only once. Therefore, single-stage networks run much faster.

Several variants of YOLO have been introduced in recent years, including YOLOV2 [11], YOLOV3 [12], YOLOV4, and YOLOV5 [13], [14], [15]. Although the new versions of YOLO are more accurate, their execution speed is not noticeably faster than the basic model. This paper presents a Yolo-based method called Fast-Yolo-Rec to address this issue. In this algorithm, the position of the vehicles is

predicted in several input frames, and in some of them, detection is used to determine the position of the vehicles. This is because predicting the situation of vehicles is done much faster than detecting them, which increases the average speed of implementing the Fast-Yolo-Rec algorithm. On the other hand, the accuracy of the presented algorithm is maintained due to the use of the detector network in this algorithm. These networks complement each other and help maintain accuracy and increase speed.

The proposed detector in Fast-Yolo-Rec uses a semantic attention mechanism. This mechanism improves vehicle positioning accuracy and reduces the target miss rates based on its superior performance. The accuracy of the proposed detector in the Fast-Yolo-Rec algorithm is comparable to the last versions of YOLO. Still, its number of parameters is less, and its speed is higher. The reason for achieving this desired result is using the meaningful attention mechanism, which is implemented using U-Net-based segmentation networks. In this mechanism, feature maps are generated so that they make a significant distinction between the vehicles and the image background. Until now, such a mechanism has not been used in YOLO family detectors. This mechanism makes vehicle positioning more accurate, and the miss rate of the target is reduced. Despite this mechanism, there is no need to deepen the network to achieve higher accuracy. Thus, many problems caused by deepening the network, such as overfitting and high hardware volume, are solved. On the other hand, the speed of the detector does not sacrifice its accuracy.

In the Fast-Yolo-Rec algorithm, motion prediction is also used in addition to detection. In motion prediction, only helpful information is processed in sequential images. Consecutive images in the traffic control system have much redundancy due to the stability of surveillance cameras and the existence of a common background in their recorded images. The reason for the high speed of prediction compared to detection is the elimination of these redundancies.

Traditional methods that use motion include optical flow and the use of differential images. [16], [17], [18]. Differential methods have several significant drawbacks. They cannot detect the position of stationary objects and are not suitable for detecting slow and fast objects. Furthermore, when the background of an image changes, they mistakenly assume the change is a moving object.

Optical flow-based detection methods calculate the direction and velocity of each pixel in an image and use them to separate the moving region from the image background [19]. They are highly dependent on the quality of the input images. Considering the mentioned problems, instead of traditional motion-based detection methods, deep neural networks are used for motion prediction, which also finds long-term dependencies and therefore has higher accuracy.

To improve the prediction accuracy, before using deep neural networks, the trajectory of each car is classified, and it is determined whether the car moves in a straight line or changes its direction to the right or left. Then, predictive networks are

performed according to the trajectory specified by the classifier. Another solution is the shortening of the prediction time. Research [20], [21], and [22] shows that shorter forecasting time increases forecasting accuracy. Therefore, in this study, prediction is done only in even frames, and detection is used in other frames. Although, according to the complexity of the scene, the time of prediction and detection can be changed and the accuracy and speed of the algorithm can be managed. In fact, the balance between accuracy and speed is one of the challenges of the detection problems that have been addressed in this study. Overall, the main contributions of the present work include:

- It proposes a flexible algorithm (Fast-Yolo-Rec) in terms of speed and accuracy to find vehicle positions. Depending on the complexity of the image and the predefined speed and accuracy, only the proposed detection network or the integration of the proposed detection and prediction networks can be used in an alternating period. In this research, these two networks are used alternatingly. The detection network is used in primary and odd frames, and the prediction network is used in even frames. Since the predicted network is faster than the detection network, the algorithm speed increased. The prediction network accuracy are improved by regularly using information from the detector in specified frames. These two networks complement each other and improve speed and accuracy.

- A Yolo-based detection network (SSAM-YOLO) that has the following advantages over high-speed one-stage detection networks:

  - It decreases the hardware requirement and accelerates detection by reducing the number of learnable parameter.
  - It improves the detection accuracy using a novel semantic attention mechanism (unlike in popular Yolo-based detection networks) and more effective feature maps where vehicles are effectively differentiated from the background.
  - The scale change robustness of the detection network is increased by using detection heads with two different scales, 13×13 and 26×26, and the multi-receptive field block (MRF block) in the backbone of the detector. Transferring feature maps of different receptive fields created by parallel paths in the MRF block facilitates the detection of objects of different sizes.

- It provides an efficient and effective algorithm for vehicle trajectory classification and vehicle position prediction based on LSTM recurrent networks and regression. It is more accurate than traditional trajectory predicting methods.

The SSAM-YOLO detection network has comparable accuracy to the last version of YOLO detectors. The baseline Highway category from the CDNet2014 dataset is used to train and evaluate the proposed detector. The proposed

detection network has almost the same level of accuracy as YOLOV4_Tiny but 29.38% fewer parameters. It also has 31% fewer parameters and 46% fewer floating point operations than YOLOV7_TINY. Therefore, it has higher speed at the same accuracy. The feature maps produced by the proposed attention mechanism, which successfully separates vehicles from the background, enable the efficient reduction of this parameter. Despite this module, the network is not deepened to obtain better features. The use of fewer parameters accelerates the algorithm and downsizes the hardware. Using the predictor and the SSAM-YOLO detector allows the proposed Fast-Yolo-Rec to take advantage of both methods simultaneously and achieve high accuracy in vehicle detection in addition to the appropriate speed. The results obtained from the implementation of the proposed method and its comparison with the baseline methods are discussed in the evaluation section.

II. RELATED WORK

In recent decades, vehicle detection has greatly interested machine vision researchers. Cameras have increasingly advanced in terms of hardware. They are today more cost-effective and widely used in traffic control systems.

There are two broad categories of vehicle detection methods: traditional and deep network-based. Models based on colors, object contours, and optical flow [23], [24] and background modeling algorithms, such as a mixture of Gaussian (MOG) and its subtraction from the input image [25], [26], HOG and Haar-like features [27], Generalized-Huff, and Kalman filter are commonly used as methods of the former group to detect and predict the positions of vehicles [28], [29]. Principal component analysis (PCA) is employed to provide more efficient data, and support vector machines (SVMs) are used to classify data [30], [31].

In the latter group of methods, features are determined automatically with the help of deep networks. The performance of such features depends on the training dataset, the type, and the effectiveness of the network. A larger number of training data and higher relevance lead to higher accuracy. Networks-based detection methods are divided into two-stage and one-stage groups. Two-stage methods, such as RCNN, are more accurate; however, they lack sufficient speed in real-time applications. They use region search [32] in the image and convolutional network, have a long training time, and require large memory. Mask RCNN, FPN, and R-FCN [33] have improved the feature extraction and classification efficiency of convolutional networks.

Single-stage networks, such as SSD [34] and YOLO, are faster and less accurate than two-stage methods. SSD utilizes MutiBox [35], Region Proposal Network (RPN), and multi-scale representation methods to more accurately locate an object. The YOLO network divides an image into a set of grids. Each grid is responsible for predicting objects whose center points are located within the grid. YOLO variants, e.g., YOLOV2, YOLOV3, and YOLOV4, have been introduced. YOLOV2 improves the YOLOV1 using Darknet-19 as the

backbone and anchor boxes to predict the bounding boxes and batch normalization, which normalizes the input of each layer and accelerates network convergence. YOLOV2_Tiny is a very efficient and effective variant in real-time applications. YOLOV3 detects objects at different scales. Thus, it is slower than YOLOV2 and has higher scale change robustness. Moreover, YOLOV4 improves YOLOV3 using CSPDarknet53. Then the concept of a decoupled head was introduced in YOLOX. It has been updated to use a decoupled head and achieve higher accuracy. There are variations of YOLOX split into two categories; Standard Models for high precision and Light Models for edge devices. YOLOX-s is able to achieve the same accuracy as YOLOv4 with half the processing time [36].

YOLO v5 uses Cross-Stage Partial Connections with Darknet-53 as the Backbone and Path Aggregation Network as the Neck, just like the YOLO v4. The significant improvements include novel mosaic data augmentation and auto-learning bounding box anchors. Based on YOLOv4, [14] proposes a YOLOV4-5D network for improving detection accuracy. The backbone network in YOLOV4-5D is the SPDarknet53_dcn(P). The last output layer in the CSPDarknet53 is replaced with deformable convolution to enhance the detection accuracy. In YOLOV4-5D, a new feature fusion module (PAN++) is designed, and five scale detection layers are used to improve the detection accuracy of small objects.

According to the benchmarking performed by Meituan's team, YOLOv6 outperforms YOLOv5 in terms of accuracy and speed. YOLOv6 uses the EfficientRep backbone. Unlike the previous YOLO architectures, which use anchor-based methods for object detection, YOLOv6 opts for the anchor-free approach. This makes YOLOv6 faster compared to most anchor-based object detectors [37]. After that, YOLOV7 was presented by Chien-Yao Wang and colleagues [38]. YOLOv7 enhances object detection by creating a network architecture that predicts bounding boxes more accurately than its competitors at comparable inference speeds. This method uses the extended efficient layer aggregation networks (E-ELAN) to achieve these results. The E-ELAN does not change the gradient transmission path of the original architecture but uses group convolution to increase the cardinality of the added features.

Several works sought to increase the accuracy of detection networks. Tinier-YOLO was developed based on Tiny-YOLO-V3 [39]. The fire module in SqueezeNet is chosen in Tinier-YOLO by looking into the number of fire modules and their locations within the model. A convolutional neural network (CMNet) was proposed for fast vehicle detection in complex scenes [40]. First, it suggests a connect-and-merge residual network (CMRN). Then, a multi-scale prediction network (MSPN) is used to accurately regress the vehicle shape and categorize different types of vehicles.

An intelligent traffic-monitoring system was developed using YOLO and a convolutional fuzzy neural network (CFNN) [41]. It logs the traffic flow on the road. It uses a vehicle-counting technique along with the detection of vehi-
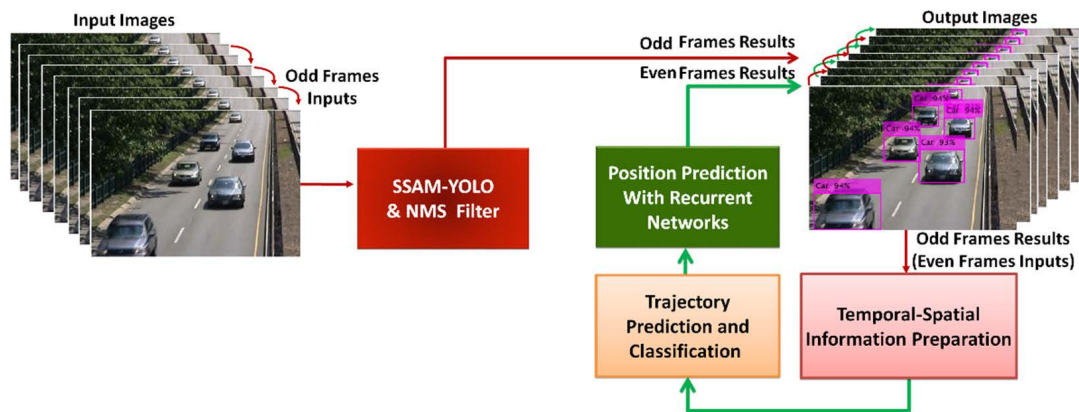
FIGURE 1. Schematic of the proposed Fast-Yolo-Rec algorithm.

cles to estimate the traffic flow. Then, two efficient models (i.e., CFNN and VectorCFNN) and a network mapping fusion method are proposed to classify vehicles.

A context-exploited method was introduced to integrate features from various receptive fields to obtain contextual representation and improve detection accuracy [42]. To encode the context, it employed multi-branch diverse receptive field module principles.

[43] proposes a Compressed Sensing Output Encoding (CSOE) for detecting pixel coordinates of small objects and crowd counting and localization. The proposed detection framework in [43] consists of a crowd location encoding scheme based on compressed sensing and an end-to-end trainable network made up of observation layers and sparse reconstruction layers and achieves excellent results in scenes with high crowd density. [44] creates an oriented surgical tool with CSLE, a new backpropagation rule for sparse reconstruction, and an end-to-end trainable network. Its approach is quite effective in casting-oriented object localization as regression in encoding signal space.

[45] proposes a dual-branch center face detector (DBCFace). This paper improves face detection via a dual-branch fully convolutional framework without extra anchor design and NMS. It uses two parallel detectors, does not rely on NMS, and achieves similar performance as anchor-based methods with multi-branch. [46] proposes a novel architecture named Serial and Parallel Group Network (SPGNet), which can capture discriminative multi-scale information while keeping the structure compact. Various computer vision tasks, such as image classification, object detection, and person re-identification, have been used to evaluate the SPGNet.

[47] proposes a co-attention scheme containing classagnostic attention (CA) and semantic attention (SA). These capture object boundary details and global context-aware information from low-level and high-level features. This model can filter out the distracting distraction of background information by fusing these two attentions.

Despite significant breakthroughs in deep learning networks for object detection, the trade-off between detection

accuracy and speed remains a challenge. The present study aims to improve the detection speed of the algorithm while maintaining an accuracy comparable to the most recent variants of high-speed detectors, like recent variants of YOLO. Our research also has the ability to locate the occluded vehicle using the pre-blocking areas, which is a benefit. The detector network cannot determine its position when a vehicle is blocked behind an obstacle, such as a larger vehicle or bus. Nonetheless, the predictive network in the proposed Fast-Yolo-Rec can quickly locate the location. Thus, it can be said that another advantage of our research is that it has the ability to locate the occluded vehicle based on the pre-blocking areas. This study handles this challenge by integrating deep learning networks, such as Yolo-based convolutional networks, classifiers, recurrent networks, and segmentation networks.

III. PROPOSED ALGORITHM

Figure 1 depicts the proposed algorithm. It uses two distinct deep networks to find vehicle positions. One deep network is implemented in odd frames (SSAM-YOLO), while the other is executed in even frames. These networks include the recurrent vehicle position prediction network and a YOLO-based detection network. The detection network uses the semantic attention mechanism and is executed in the first 64 frames of consecutive images and odd-numbered frames. As prediction is faster than detection, using a prediction block accelerates the algorithm. In even frames, time-series data should be provided for the trajectory prediction network before using the prediction network. These accurate data are obtained from the detection network in odd frames and increase the accuracy of the prediction network. A rise in the prediction time usually causes errors. The results in this study depict that for a prediction time below 28 frames, the position prediction error is below two pixels, which is desirable. Frames are used alternatingly in the prediction network, and the prediction time of one frame has a very good accuracy.

Therefore, the proposed algorithm has a high speed without losing accuracy, which is not the case with even the fastest detectors. In addition to the time of prediction shortening to
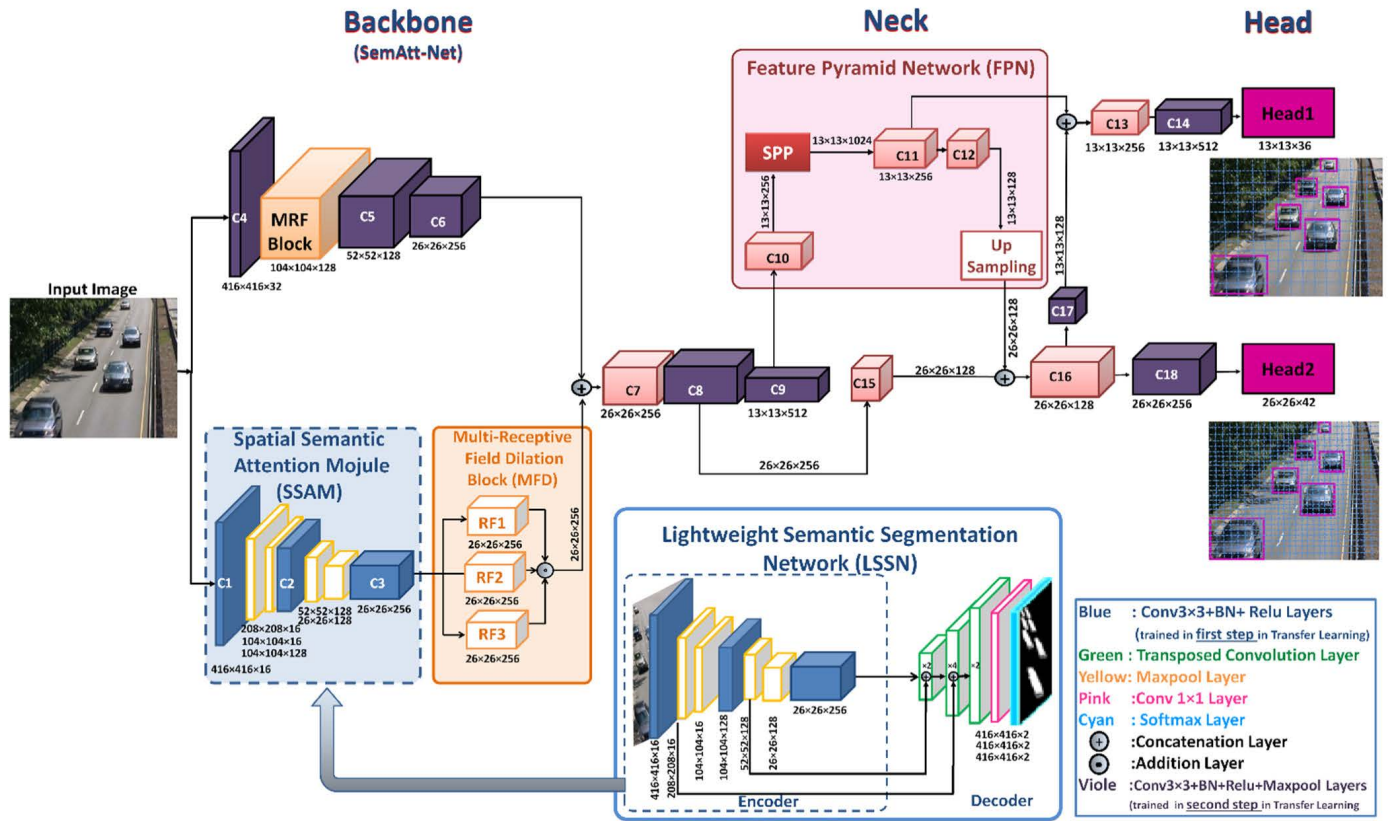
FIGURE 2. Schematic of the proposed SSAM-YOLO detector in Fast-Yolo-Rec algorithm.

increase accuracy, the vehicle trajectory is identified using a classification network. Then, according to the recognized trajectory, the position of the vehicle is determined through the prediction network trained for that trajectory. The classification network identifies whether the vehicle moves straight or takes a lane change. Then, based on the direction identified by the classifier, the recurrent network trained for the corresponding direction is used.

A. DETECTION NETWORK
SSAM-YOLO is proposed as a Yolo-based detection network, as shown in Figure 2. The backbone, neck, and head are the three main components of YOLO-based detectors.

Due to their higher resolution and more accurate spatial features, the extracted feature maps of the backbone are more effective for vehicle detection than for other feature maps in the detection network. The head and neck are more helpful in classifying vehicles since they provide higher semantic data and depth, despite lower spatial detail due to lower resolution. This paper proposes a design for the SSAM-YOLO detector that improves the accuracy of vehicle position detection and increases scale change robustness in light of the MRF block and SSAM module in the backbone, referred to as the Semantic Attention Network (SemAtt-Net).

The MRF Block is constructed to improve feature map extraction at a 104*104 resolution with various receptive fields. In this block, residual connections are used. A structural comparison based on the residual connections between multiple blocks is shown in Figure 3. Figure 3(a) indicates the residual-based block in the YOLOV3 detector, in which two series residual branches are used. Figure 3(b) illustrates the utilization of two residual branches in the CSP blocks of the YOLOV4 detector. Figure 3(c) depicts the structure of the proposed MRF block. The backbone of YOLOV4_Tiny consists of three CSP blocks. The input and output of this block are concatenated by two connections, one of which is the residual connection, and the other one is composed of a slice layer, two 3×3-layer convolutions, and a layer with a factor of 1×1 convolutions. In the second connection, the layers are successive, and the output attribute maps of the CSP block in this connection are calculated in the 5×5 received field.

The proposed MRF block includes one residual connections, two 3×3 convolution blocks, one atrous convolution block, and one 1×1 convolution block. The extracted feature maps in the MRF block are concatenated into a 3×3 and a 5×5 receptive field and transferred to the following convolutional layers. Feature maps with a 3×3 receptive field and those with a 5×5 receptive field in the previous blocks are transferred to the next layer through the MRF block used instead of the earlier blocks.

Thus, feature maps include more spatial details due to their smaller receptive field. The atrous layer also considers
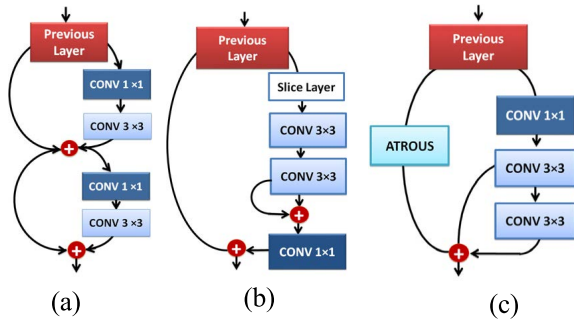
FIGURE 3. Comparison of blocks with residual branches between (a) two residual blocks in the sequence in YOLOV3, (b) a CSP block in YOLOV4 with two residual blocks, and (c) proposed MRF block.

FIGURE 4. Comparison of segmentation networks for (a) the input image, (b) complete segmentation network, and (c) proposed LSSN network.

another 5×5 receptive field with different detail. Therefore, the proposed detector effectively detects vehicles of different scales, a challenge in conventional detection networks.

Apart from the MRF block, the SSAM module is used to improve feature maps in the backbone of the SSAM_YOLO detector. The SSAM module in SSAM_YOLO is a UNET-based semantic segmentation network encoder known as LSSN, which was very lightly designed in terms of the number of parameters. This network helps the detector generate desired feature maps in its backbone. Vehicles in these maps are well distinguished from the background, increasing the precision of the detector.

LSSN receives independent training (first training stage). The encoder is then employed as the SSAM module in the detection network. The SSAM module in the detection network pays attention to the spatial positions of vehicles in feature maps with a resolution of 26×26 and effectively distinguishes between (foreground) and background vehicles. The detection and LSSN segmentation networks are trained using the same images. As a result, SSAM_YOLO can be trained using transfer training. Also, when training SSAM_YOLO, the learning rate in the SSAM module is zeroed, and the remaining detector training is carried out (second training stage).

As mentioned, the LSSN network is designed and trained to use its encoder in the SSAM_YOLO detector; however, feedback from the encoder to the decoder in LSSN decelerates the detector. Hence, such feedback is eliminated in the SSAM module. To reduce the feedback dependence of trainable parameters in LSSN, minimal feedback is used. Moreover, to decrease the number of SSAM module parameters and accelerate the detector, convolution layers at resolutions 52×52, 104×104, and 208×208 were eliminated from the complete UNET network, utilizing only the pooling layer to obtain a lower resolution.

Figure 4 compares the proposed light semantic segmentation network (LSSN) to a complete segmentation network (CSN) with more feedback and convolution layers in output. As can be seen, the reduction of feedback and convolution layers decreases segmentation accuracy on foreground boundaries in the output of the LSSN network; however,
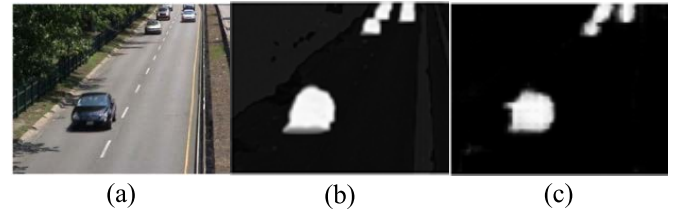
boundary accuracy in the final layer of LSSN with a resolution of 416×416 has no significant effect on the output of the SSAM module with a resolution of 26×26 (due to high pooling); But the number of parameters and network feedback quantity in LSSN is dropped dramatically, and the use of its encoder in the backbone of the detector accelerates detection.

In the output feature maps, only the central pixels of the cars are separated from the background. Therefore, the MFD block is designed to pay more attention to the cars in the feature maps. The MFD block consists of three maximum pooling layers with receptive fields 2×2, 3×3, and 4×4 (RF1, RF2, and RF3 in Figure 2). This block uses multiple receptive fields for maximum pooling with stride $= 1$ (no pooling) and has a dilation-like function to ensure that full attention is paid to pixels corresponding to vehicles. More attention improves vehicle detection accuracy and reduces target miss rates.

Figure 5(a) depicts a 416×416 input image of the SSAM_YOLO network, while Figs. 5(b)-5(d) show the output feature maps of the convolution layer C6, SSAM module, and MFD block shown in Figure 2. The foreground (vehicles) is effectively differentiated from the background in Figs. 5(c)-5(d).

As can be seen, the proposed SSAM module, MFD block, and transfer learning used to train the SSAM_YOLO detection network provide more effective features than standard Yolo detector convolution layers (figure 5(b)). Finally, concatenating the feature maps from figure 5(d) to the feature maps from figure 5(b) and applying them to the next layer improves detection accuracy.

In this study, other factors improve vehicle detection accuracy, including the selection of suitable anchors. YOLO can function efficiently when several objects are associated with one grid cell. However, in the case of an overlap, where one grid cell contains two different objects, anchor boxes can be used to enable one grid cell to detect several objects. To increase detection accuracy in the proposed SSAM_YOLO detector, the number and size of anchors were chosen more accurately from a different perspective than in other detectors.

Typically, the number of anchor boxes is determined by the number of object classes detected. In this study, 11 anchors were chosen based on the mean intersection over union to have greater scale robustness in detection.

416 × 416   26×26   26×26   26×26

(a)   *(b)*   *(c)*   *(d)*
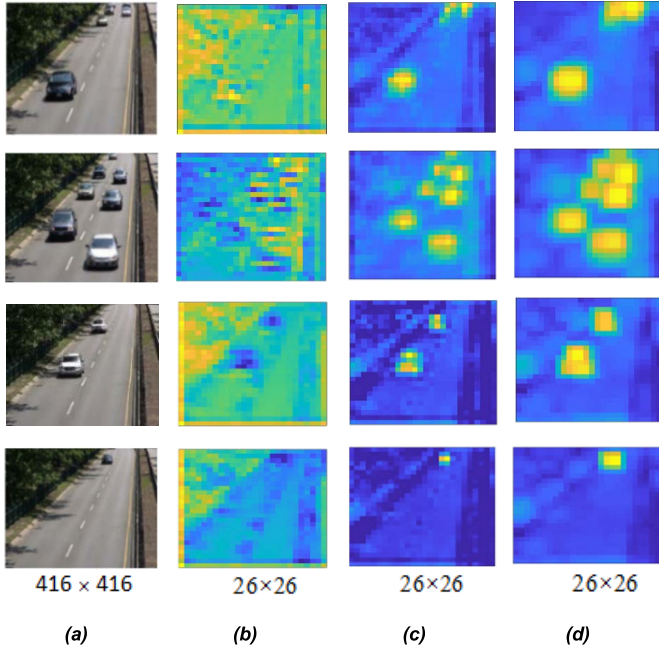
FIGURE 5. Comparison of feature maps for (a) input image, (b) output of convolution layer C6 (Figure 2), (c) output of the proposed SSAM module, and (d) output of the proposed MFD block.

B. TEMPORAL-SPATIAL INFORMATION PREPARATION

The position prediction block uses the history of vehicle movement as time series data and predicts the position of the vehicle in even frames according to it.

$$X = [X^{(t-t_h)}, \ldots, X^{(t-1)}, X^t] \tag{1}$$

where t_h is a fixed (history) time horizon,

$$X^t = [x_0^{(t)}, y_0^{(t)}, x_1^{(t)}, y_1^{(t)}, \ldots, x_n^{(t)}, y_n^{(t)}] \tag{2}$$

where x and y are the vehicle coordinates at time t. Since there are several vehicles in each image, a specific name or unique label should be assigned to each vehicle as long as the vehicle is in the surveillance camera field of view so that the data of each vehicle are stored in a dedicated sub-tensor for the exact vehicle. Let $[x_0^{(t-t_h)}, y_0^{(t-t_h)}, \ldots, x_0^{(t)}, y_0^{(t)}]$ be the position history of the first vehicle and $[x_n^{(t-t_h)}, y_n^{(t-t_h)}, \ldots, x_n^{(t)}, y_n^{(t)}]$ be the position history of vehicle n. The area history of vehicles is $[A_0^{(t-t_h)}, \ldots, A_0^{(t)}]$ to $[A_n^{(t-t_h)}, \ldots, A_n^{(t)}]$, and the intensity average history is $[I_0^{(t-t_h)}, \ldots, I_0^{(t)}]$ to $\left[I_n^{(t-t_h)}, \ldots, I_n^{(t)}\right]$. The similarity distance criterion (SDC) for a vehicle with features $[x^{(t+1)}, y^{(t+1)}, I^{(t+1)}, A^{(t+1)}]$ can be defined as:

$$DPOS0 = \left(x^{(t+1)} - x_0^{(t)}\right)^2 + \left(y^{(t+1)} - y_0^{(t)}\right)^2 \tag{3}$$

$$DAR0 = \left(A^{(t+1)} - A_0^{(t)}\right)^2 \tag{4}$$

$$DIN0 = \left(I^{(t+1)} - I_0^{(t)}\right)^2 \tag{5}$$

$$D^0 = \sqrt{\alpha (DPOS0) + \beta(DAR0) + \gamma(DIN0)} \tag{6}$$

$$DPOSN = \left(x^{(t+1)} - x_n^{(t)}\right)^2 + \left(y^{(t+1)} - y_n^{(t)}\right)^2 \tag{7}$$

$$DARN = \left(A^{(t+1)} - A_n^{(t)}\right)^2 \tag{8}$$

$$DINN = \left(I^{(t+1)} - I_n^{(t)}\right)^2 \tag{9}$$

$$D^n = \sqrt{\alpha (DPOSN) + \beta(DARN) + \gamma(DINN)} \tag{10}$$

$$SDC = [D^0, \ldots, D^n] \tag{11}$$

$$label = \left\{ \begin{array}{l} \arg (SDC) \min (SDC) < Th \\ n + 1 \min (SDC) \geq Th \end{array} \right\} \tag{12}$$

where D^0 is the similarity distance with the first vehicle seen in the scene, D^n is the similarity distance with the vehicle n in the background, and SDC contains all of them. A vehicle label is determined by the minimum value of the SDC and its index. As can be seen, the similarity distance between two vehicles is calculated by the square of the distance between the positions, the average intensity, and their area. Here, α, β, and γ are hyper parameters.

Figure 6 illustrates the labelling output of several sequential frames from the Highway dataset. As can be seen, labelling is stable, and the label of each vehicle remained unchanged in consecutive frames. According to these labels, the position of each vehicle in consecutive frames is recorded as the history of the movement of that vehicle specifically for it. Then, it is recorded in the FIFO-like temporal tensor and used in the prediction block.

C. POSITION PREDICTION AND TRAJECTORY CLASSIFICATION BLOCKS

In addition to designing a fast and accurate SSAM_YOLO detection network, the position prediction network is presented in this study. The goal of this predictor is to maximize the vehicle position detection speed. Unlike traditional motion prediction methods, LSTM-based recurrent networks account for long-term time dependencies and are thus more accurate than conventional position prediction models, e.g., the constant acceleration (CA) model. The position of a vehicle in the current frame (p_2) is compared to that of the vehicle in the previous frame (p_1) in this model.

$$p_2 = \frac{1}{2}a\Delta t^2 + v\Delta t + p_1 \tag{13}$$

where acceleration a and velocity v are assumed to be known. However, this assumption is not always the case, and the speed and acceleration of vehicles may change many times, depending on traffic flow and driver. In recurrent networks, redirection is learned by the network for different modes of speed and acceleration over time. Since they contain several nonlinear activation functions, they could predict complex movement patterns and trajectories with different accelerations and velocities. These functions determine the data points of the previously saved frames that should be kept or excluded and the data of the current frame that should be added. Assuming h_{t-1} and X_t to be the inputs, the operation of the LSTM network can be formulated as:

$$i_t = \sigma (W_i X_t + U_i h_{t-1}) \tag{14}$$
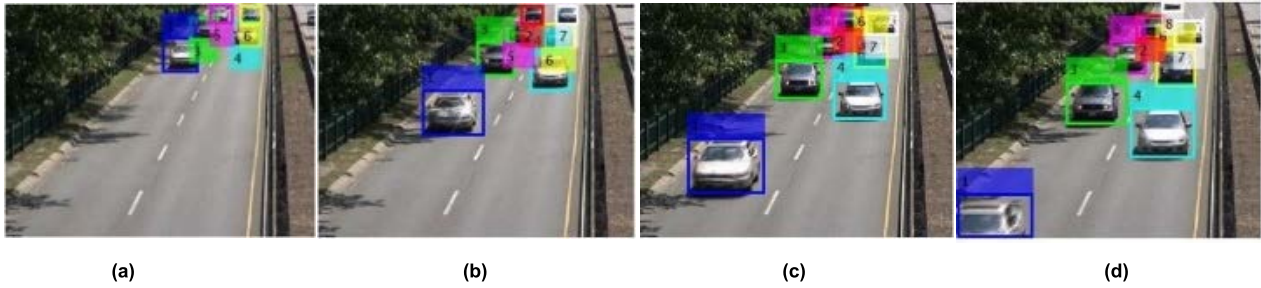
FIGURE 6. Visual results of the proposed stable labelling on consecutive images with different time interrupt ((a) to (d)).
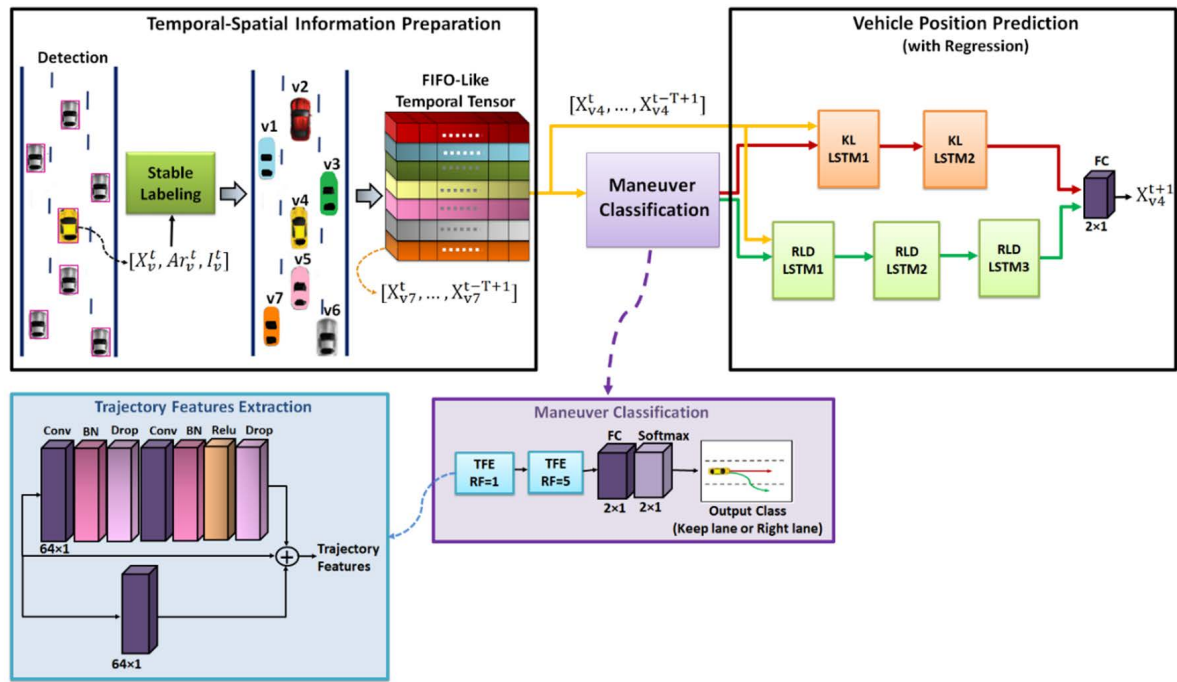
FIGURE 7. Proposed vehicle position prediction architecture.

$$f_t = \sigma(W_f X_t + U_f h_{t-1}) \tag{15}$$

$$o_t = \sigma(W_o X_t + U_o h_{t-1}) \tag{16}$$

$$c_t = \tanh(W_s X_t + U_s h_{t-1}) \tag{17}$$

$$s_t = i_t \odot c_t + f_t \odot s_{t-1} \tag{18}$$

$$h_t = o_t \odot \tanh(s_t) \tag{19}$$

where h_{t-1} is the network output in the preceding frame, while X_t stands for the data introduced in the current frame v.

It contains the historical trajectory $[X_v^{t-T+1}, \ldots, X_v^t]$ of the vehicle obtained from the time series data preparation block. Also, "$\odot$" denotes the dot product symbol.

Weight matrices W_i, U_i, W_f, U_f, W_0, and U_0 represent the input, forget, and output gates, respectively, h_t is the network output, and state cell s_t is updated in each frame. The LSTM network is employed to regress vehicle trajectories in two modes of lane keeping and right turns in the Highway dataset. The proposed algorithm architecture is depicted in Figure 7. It includes sections on temporal-spatial information

preparation (TSIP), trajectory classification (TC), and vehicle position prediction (VPP). Subsection 3.2 described the TSIP block. The vehicle trajectory is determined in the TC section, which is designed to improve prediction performance.

The classifier determines whether the vehicle is moving in a straight line or changing lanes. The trajectories of vehicles may differ. Given that in this study, there are two categories of straight trajectory and rightward lane change in the dataset used to train recurrent networks, a classifier with two classes is proposed, and KL-LSTM or RLD-LSTM networks are employed to predict vehicle positions based on the classifier-identified trajectory class.

Trajectory classification is carried out by sampling various trajectories learned for each trajectory. It consists of two trajectory feature extraction (TFE) blocks with different receptive fields, a fully connected layer, and a soft-max layer.

The TFE block has three parallel paths. In the first path, the convolution layer is initially used. Let x be the input of the TFE block, and convolution filters are determined using

weights w_1 and bias b_1. The first convolution layer output feature map (f_{s1}^1) is written as:

$$f_{s1}^1(x; w_1, b_1) = w_1 *_{s1} x + b_1 \tag{20}$$

where $*_{s1}$ is the convolution operation with stride s1.After the convolution layer, the batch normalization (BN) layer and drop-out layer are exploited. BN plays a key role in avoiding vanishing gradients. For B1 representing a batch of feature maps, BN is defined as:

$$B_1 = batch\left(f_{s1}^1\right) \tag{21}$$

$$\hat{B}_1 = \frac{B_1 - \mu_{B1}}{\sqrt{\sigma_{B1}^2 + \varepsilon}} \tag{22}$$

$$D_1 = Drop\left(\hat{B}_1\right) \tag{23}$$

where $\hat{B}_1$, σ_{B1}, and μ_{B1} are the normalized array, variance, and mean of batch B1, respectively. The value of ε is set to 0.001 to prevent null point division. After normalization, the dropout layer is used. The dropout layer is a mask that nullifies the contributions of some neurons to the next layer and leaves others unmodified. Then, another convolution layer is used, the input of which is the dropout output, and convolution filters are determined using weights w_2 and bias b_2. The output feature map f_{s1}^2 is defined as:

$$f_{s1}^2(D1; w_2.b_2) = w_2 *_{s2} D1 + b_2 \tag{24}$$

where $*_{s2}$ is the convolution operation with stride s_2. Then, the batch normalization (BN) layer, ReLU activation layer, and dropout layer are used.

$$B_2 = batch\left(f_{s1}^2\right) \tag{25}$$

$$\hat{B}_2 = \frac{B_2 - \mu_{B2}}{\sqrt{\mu_{B2}^2 + \varepsilon}} \tag{26}$$

$$R = \max\left(0, \hat{B}_2\right) \tag{27}$$

$$D_2 = Drop(R) \tag{28}$$

The use of ReLU helps prevent exponential growth in computation required to operate the classifier network and introduce non-linearity into the BN layer output. The second path in the TFE block is a residual connection used to improve network learning. The third path is a convolutional layer whose receptive field is different in the two TFE blocks; RF=1 in the first block and RF=5 in the next block, while the receptive fields of the first path are the same for the two TFE blocks. The output feature map f_{s1}^3 is defined as:

$$f_{s2}^3(x; w_3, b_3) = w_3 *_{s3} x + b_3 \tag{29}$$

Finally, the feature maps from the first and third paths and the input of the TFE block are combined to form the final feature map out_{MRF}.

$$out_{MRF} = \sum_{i=0}^{N}(x(i) + D_2(i) + f_{s2}^3(i)) \tag{30}$$

The obtained properties are transferred through a fully connected layer fc_MRF to the Softmax layer. This layer determines the probability of each of the two trajectory classes.

$$S^{(fc\_MRF_k)} = \frac{\exp(fc\_MRF_k)}{\sum_{k=1}^{N}\exp(fc\_MRF_k)} \tag{31}$$

where N is the length of fc_{MRF} (i.e., 64). After trajectory classification, VPP is executed, where there are two groups of LSTM networks: KL_LSTM and RLD_LSTM. KL_LSTM networks are executed when the straight-line trajectory (Keep-Lane) class is specified in TC, and RLD_LSTM networks are selected when the redirection mode is set to the right. The trajectory is non-linear and more complex in the latter, and RLD_LSTM networks are more than KL_LSTM networks.

IV. RESULTS AND DISCUSSION

This section evaluates the proposed algorithm using Highway data from the CDNet2014 dataset. The algorithm consists of several DNN networks, including the SSAM_YOLO detection network and the lightweight semantic segmentation network LSSN, designed to prepare the SSAM module in SSAM_YOLO. Also, the KL_LSTM trajectory classifier network and RLD_LSTM are used in the algorithm. These networks were trained using the adaptive moment estimation optimization algorithm. The performance of the proposed algorithm is evaluated through comparison to previous works in the average precision (AP), average execution time, and RMSE. The tests were conducted in MATLAB with GPU computing facilities on a PC with a Core i7–3.60GHz CPU, 16GB RAM, GTX1060 GPU, and a Microsoft WINDOWS 10-64bit OS.

A. DATASET

In this study, several deep networks are trained, some of which, i.e., SSAM_YOLO and LSSN networks, are used in odd frames, while the others, i.e., KL_LSTM, RLD_LSTM, and trajectory classification networks, are used in even frames. Due to the design of the SSAM module in the detector, which is responsible for dividing the image into two classes, i.e., vehicles and background, the detection network training requires a dataset suitable for object detection and segmentation tasks.

The KL_LSTM and RLD_LSTM prediction networks are used in even frames, and their input is time-series data that includes the positions of vehicles in consecutive images. Thus, a training dataset of consecutive images is required. And for this reason, the accuracy obtained from our tests is higher and closer to each other than similar references that have used datasets that include scattered and non-sequential images. The Highway dataset from the CDNet2014 dataset contains 1700 consecutive images captured by Highway surveillance cameras, and it also has good ground-truth data for image segmentation. As a result, it is suitable for training the LSTMs and LSSN networks used in this study. However, it does not have relevant data to train the detector. Therefore,

the Video Labeler of MATLAB is used to handle the ground-truth challenge of the detector. It enables labelling of the ground-truth data in an image sequence. This app can define rectangular regions of interest (ROI) labels.

B. EVALUATION METHODS

The Highway dataset was used to train the detection network. The performance of the detection network was then assessed using standard detection network evaluation metrics, i.e., AP and average execution time. For various recall levels, AP measures precision. Precision and recall are the two criteria that were used. The precision criterion represents the percentage of true positives. The recall criterion calculates the ratio of true positives to all possible outputs:

$$Precision = \frac{TP}{TP + FP} \tag{32}$$

$$Recall = \frac{TP}{TP + FN} \tag{33}$$

A true positive (TP) is an outcome where the model correctly predicts the positive class of the detected vehicles. Similarly, false positive (FP) and false negative (FN) are the outcomes where the model incorrectly predicts the positive and negative classes, respectively. The performance of the LSTM-based prediction network is evaluated using RMSE:

$$RMSE = \sqrt{(\frac{1}{N}) \sum_{j=1}^{N} (\hat{x}_j - x_j)^2 + (\hat{y}_j - y_j)^2} \tag{34}$$

where N denotes the number of vehicles in the training set. $[\hat{x}_j^i, \hat{y}_j^i]$ is the predicted position of the vehicle at time step j., and $[x_j^i, y_j^i]$ is the actual position at time step j. These networks were trained for vehicle position prediction on Highway through regression. The average execution time of the Fast-Yolo-Rec algorithm was used to measure the speed of the algorithm in sequential frames.

C. EVALUATION RESULTS

Networks were trained in the Fast-Yolo-Rec algorithm with a batch size of 4 and a total of ten epochs. The learning rate for the first half of the epochs is 0.001 and for the second half of them is 0.0001. This tutorial was additionally optimized using the SGD and Adam algorithms. Figure 8 shows precision, recall, and logarithmic average miss rate to assess the performance of the detection network.

Figure 9 depicts the performance of the detection algorithm for the Highway dataset. The proposed detector considers only the perimeter box with the highest score using the Non-Maximum Suppression (NMS) filter for vehicles with two or more perimeter boxes. The results show that the proposed method is efficient and effective. The MFD block offers improved performance with emphasis on features of the SSAM module, a dilation-like function, transfer learning in SSAM_YOLO detector training, and an optimal number of

(a)

(b)
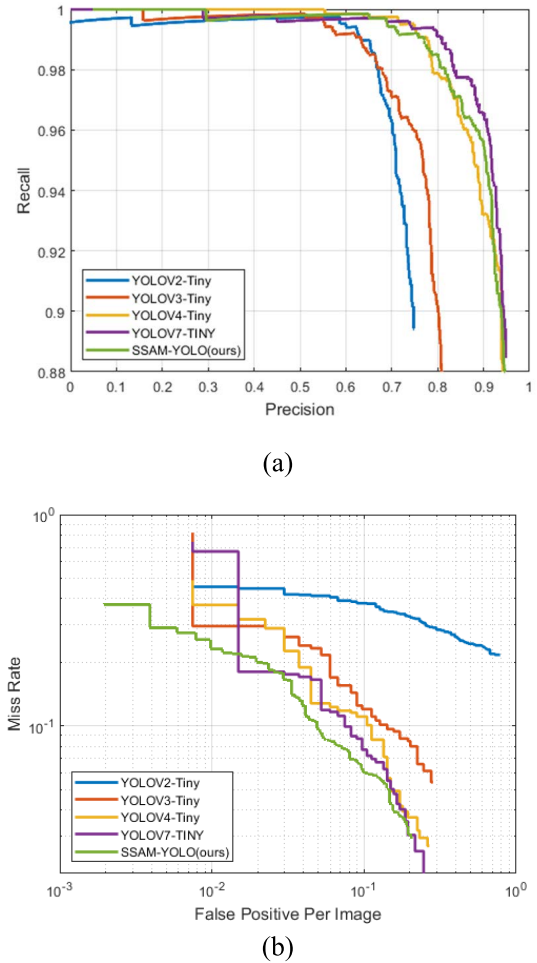
FIGURE 8. Comparison of Yolo detectors and SSAM_YOLO in terms of (a) precision, recall, and (b) miss rate.

anchors. Table 1 compares the proposed detector with variants of Yolo-based detection methods on consecutive images in the Highway dataset with one vehicle class. SSAM_YOLO has comparable accuracy to other detectors. As can be seen, its accuracy is almost equal to that of YOLOV4_Tiny and nearly 29.38% fewer parameters than YOLOV4_Tiny.

The accuracy of YOLOV7_TINY is only 1.18% higher than the proposed SSAM_YOLO detector, while it has a significantly higher computational cost (31% more parameters and 46% more floating point operations). Therefore, the proposed detector has less complexity and more effectiveness. In addition, SSAM_YOLO has a lower target miss rate than other detectors. The output feature maps of the SSAM module show the effectiveness of this module well. On these maps, the vehicle is well distinguished from the background. Distinguishing between target and background is extremely useful for network positioning and lowering miss rates. This implies that the proposed detector has fewer learnable parameters than others, leading to faster training, lower hardware demand, and higher cost-effectiveness. In light of this advantage over YOLOV4_Tiny and being comparable to YOLOV7_TINY, the proposed detector outperforms
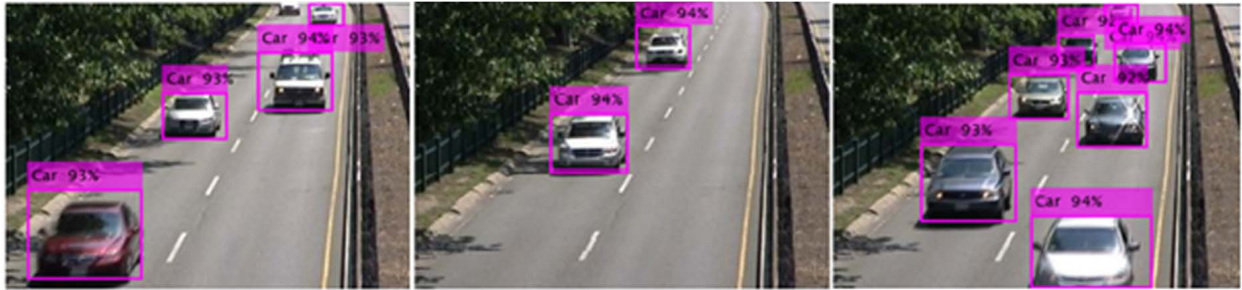
FIGURE 9. Visual results of the SSAM-YOLO detection network on Highway dataset.

TABLE 1. Performance comparison of the proposed YOLO-based detection network and earlier YOLO detection network variants for consecutive images on the Highway (baseline category of CDNet2014).

Yolo-base Detection Methods	Backbone	Input Size	Multi Scale	Attention Mechanism	Average Precision	Parameters	Miss Rate	FLOPs $\times 10^9$
YOLOV2[11]	Darknet19	416	False	False	0.7910	38.73M	0.1567	29.5
YOLOV3[12]	Darknet53	416	True	False	0.8714	49.19M	0.0980	65.8
YOLOV2_TINY [48]	FCCL	416	False	False	0.7420	7.47M	0.344	5.4
YOLOV3_TINY [49]	FCCL	416	True	False	0.8400	7.56M	0.1265	5.5
YOLOV4_TINY [50]	CspDarknet-Tiny	416	True	False	0.9400	6.06M	0.0875	4.3
YOLOX_TINY [36]	CSPDarkNet-Tiny	416	True	False	0.9412	5.06M	0.0740	6.45
YOLOV7-TINY [38]	E-ELAN-based	320	True	False	0.9524	6.2M	0.0700	5.8
SSAM-YOLO (ours)	**SemAtt_Net**	416	**True**	**True**	**0.9406**	**4.28M**	**0.0695**	**3.1**

TABLE 2. Ablation study of the proposed method in terms of average precision (AP) and miss rate.

	AP (%)	Miss-Rate (%)
Detector	88.22	15.14
Detector + SSAM (LSSN)	92.14	9.88
Detector + SSAM +MRF	93.36	7.20
Detector + SSAM +MRF+MFD	94	6.95

YOLOV2 and YOLOV3. The YOLOV2 detector has one head of detection, and the YOLOV3 detector has two heads.

Table 1 also shows that the proposed detector requires 74%, 77%, 39% and 46%fewer floating-point operations (FLOPs) than YOLOV2_TINY, YOLOV3_TINY, YOLOV4_TINY and YOLOV7 TINY, respectively. Table 2 compares the performance of the proposed SSAM-YOLO detector's SSAM module, MRF, and MFD blocks to assess the impact of each detector module on the final output.

The SSAM module improves average precision and miss rate by 3.92% and 5.26%, respectively, while the MRF block improves by 1.22% and 2.68%, and the MFD block improves by 0.64% and 0.25%. Table 2 shows that the combination of three modules in the detector Network achieves an average precision and miss rate of 94% and 6.95%, respectively.

The first five consecutive convolution layers employed by YOLOV2_Tiny and YOLOV3_Tiny are referred to as FCCL Table 1.

TABLE 3. Number of parameters in the first five consecutive convolution layers (FCCL).

Layers	Channel Input	k	Channel Output	Parameters Number
Conv1	3	3	16	448
Conv2	16	3	32	4640
Conv3	32	3	64	18496
Conv4	64	3	128	73856
Conv5	128	3	256	295168
Conv6	256	3	512	1180160

Table 3 reports the number of parameters in various FCCL layers.

In addition to achieving the desired detection accuracy through the SSAM_YOLO detector, vehicles position determination was accelerated through (1) the SSAM_YOLO detector by decreasing the number of parameters and the hardware demand compared to detectors of similar
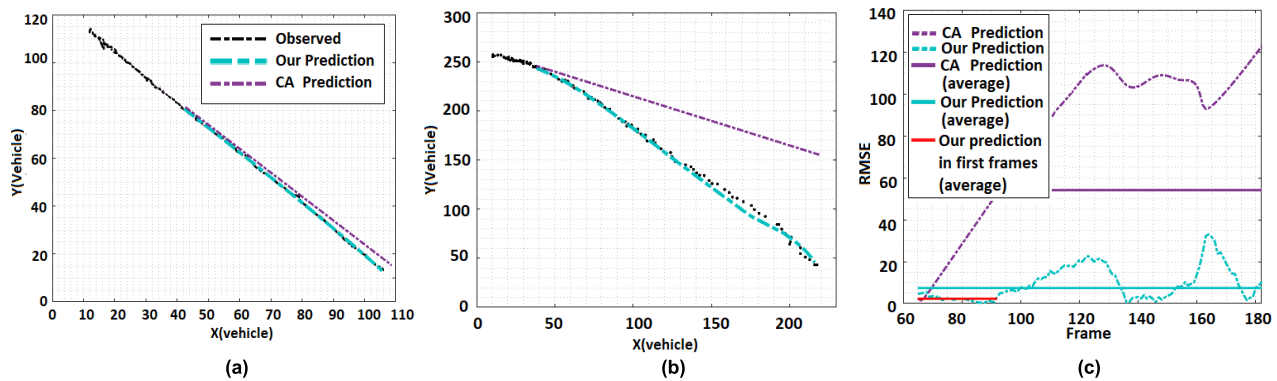
FIGURE 10. Trajectory prediction of two vehicles using the proposed and CA models (the first 64 positions in the trajectories were used for trajectory prediction); (a) the vehicle is driving straight, (b) the vehicle is taking a rightward lane change right, (c) RMSE for the proposed model versus CA model.
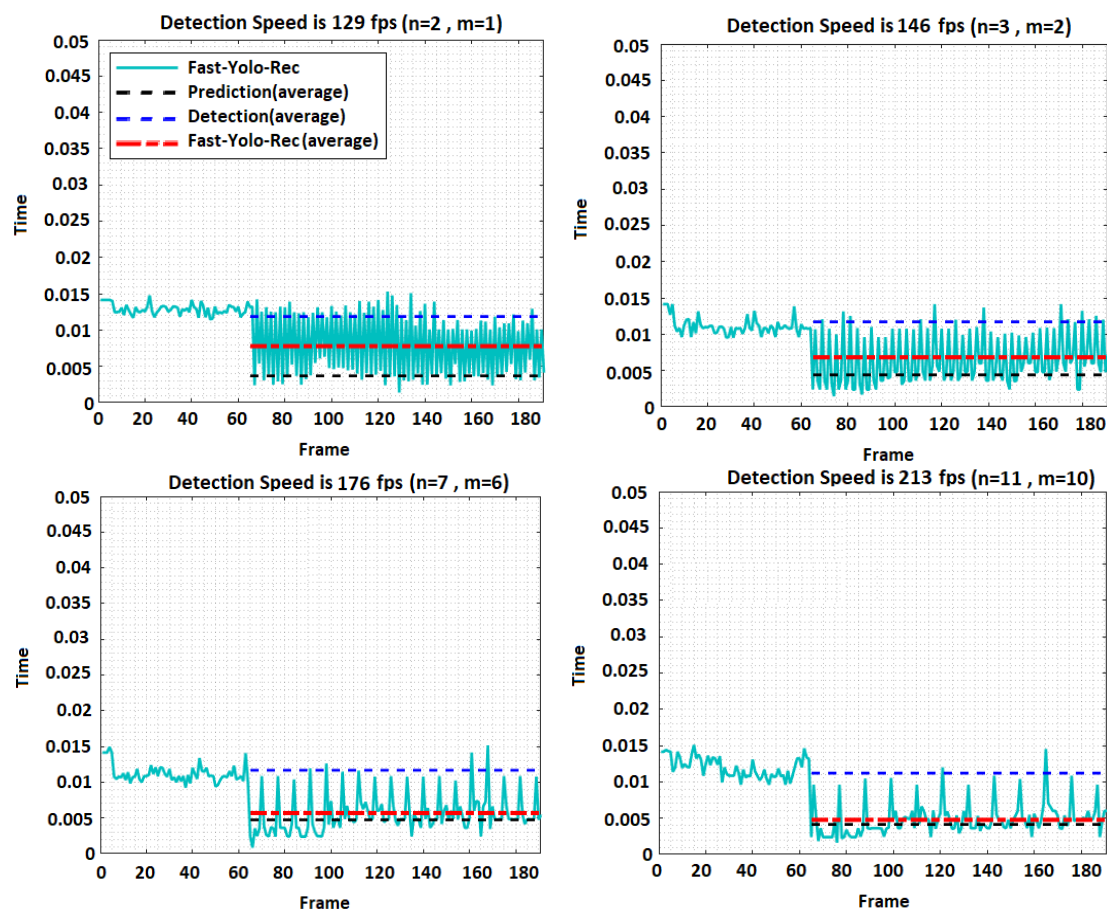
FIGURE 11. Execution time comparison of the proposed Fast-Yolo-Rec algorithm for different n-frame periods at m predictions and n-m detections of SSAM-YOLO.

performance and (2) the periodic prediction using the recurrent networks. The proposed Fast-Yolo-Rec algorithm uses trajectory classification and recurrent networks for vehicle position prediction. As shown in Figure 10, the actual trajectories of two independent vehicles are displayed in black, the predicted position of the proposed model in this study is shown in cyan, and the position prediction of the CA model is indicated in violet. In Figure 10(a), the vehicle moves in a straight line; in Figure 10(b), the vehicle changes lanes to the right. As can be seen, the trajectories are effectively predicted, and the proposed model is more accurate than the CA model. The higher accuracy of the proposed model stems from the LSTM networks and trajectory classification.

The LSTM networks consider long-term time dependencies, in contrast to the CA model. Figure 10(c) depicts the RMSE for a more accurate comparison of the two models.

As can be seen, the prediction error of the proposed model is lower than that of the CA model. The prediction implemented in successive frames without using detector data in alternating frames to compare the two models more rationally. The average RMSE in the proposed Fast-Yolo-Rec algorithm is 50% lower than the RMSE displayed in Figure 10(c) due to using the results of proposed detector with excellent accuracy in odd frames. The results of the proposed algorithm for an *n-frame* period are shown in Figure 11, while prediction was performed *m* times and detection was performed *n-m* times in an *n-frame* period. As can be seen, the prediction and detection networks shortened the average time of the proposed algorithm with a negligible RMSE, as shown in Figure 10 (c). Hence, the accuracy of the algorithm does not undergo a significant change, while detection speed increases significantly.

The red line in Figure 10 (c) shows the average prediction RMSE on the first 28 frames of the prediction. As can be seen, the detection accuracy of the first 28 frames is better than the subsequent frames, and the RMSE error is lower for them. Therefore, prediction accuracy is higher for $n \leq 28$.

The algorithm was executed for $n=2$ and $m=1$ at 129 fps. In this state, the proposed SSAM_YOLO detection network was executed in odd frames and prediction networks was performed in even frames each at this speed. The algorithm was executed at 146, 176, and 213 fps for ($n=3$ and $m=2$), ($n=7$ and $m=6$), and ($n=11$ and $m=10$), respectively.

V. CONCLUSION

The proposed Fast-Yolo-Rec technique accelerates vehicle position detection. It consists of two main parts: 1) vehicle detection network. 2) a maneuver classifier and networks for predicting vehicle position that use an alternating cycle with a certain number of frames. The detector network is used in some frames, and the classifier and predictor networks are used in other frames of this cycle. The accuracy of the Fast-Yolo-Rec algorithm has improved as the increasing the accuracy of the SSAM_YOLO detector, and it has accelerated in light of the vehicle position prediction network. As prediction is faster than detection, the use of detection in the first 64 frames and odd frames and the use of the predictor in even frames of the input images enhances the average speed of the algorithm. Moreover, by using a vehicle trajectory classifier, the accuracy of the prediction network is boosted compared to traditional vehicle position prediction approaches and LSTM networks that are particularly trained based on the output of the classifier.

The accuracy of the SSAM_YOLO detector in the Fast-Yolo-Rec algorithm is also increased through the MRF block, attention mechanism in the LSSN network and SSAM module, MFD block, transfer learning in the training of the SSAM_YOLO network, and the optimal number of anchors. An image contains several vehicles, and the data for each vehicle should be stored in a dedicated array for the same vehicle. Thus, stable labelling is required to obtain the data for the position prediction network. This data includes the

vehicle positions over time. This is fulfilled by using the detection network in odd frames, and no extra processing is required. The proposed detection and prediction networks are complementary and assist in finding the positions of vehicles faster than well-known high-speed detectors, leading to sufficient accuracy, speed, and greater flexibility. The flexibility of the proposed Fast-Yolo-Rec algorithm arose from the changing frequency of using the aforementioned networks in a periodic cycle. The real-time vehicle detection performance of the proposed algorithm is demonstrated using a real-life Highway dataset.

REFERENCES

[1] Z. Li and D. Hensher, "Understanding risky choice behaviour with travel time variability: A review of recent empirical contributions of alternative behavioural theories," *Transp. Lett.*, vol. 12, no. 8, pp. 580–590, Sep. 2020.
[2] X. Chen, Y. Chen, and G. Zhang, "A computer vision algorithm for locating and recognizing traffic signal control light status and countdown time," *J. Intell. Transp. Syst.*, vol. 25, no. 5, pp. 533–546, Sep. 2021.
[3] D. Ding, J. Tong, and L. Kong, "A deep learning approach for quality enhancement of surveillance video," *J. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 304–314, May 2020.
[4] N. Mahmoud, M. Abdel-Aty, Q. Cai, and J. Yuan, "Estimating cycle-level real-time traffic movements at signalized intersections," *J. Intell. Transp. Syst.*, vol. 26, no. 4, pp. 400–419, Jul. 2022.
[5] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.
[6] C. G. Manuel, T.-M. Jesus, L.-B. Pedro, and G.-G. Jorge, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," *Remote Sens.*, vol. 13, no. 1, p. 89, 2020.
[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[8] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2015.
[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[11] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognition*, Jul. 2017, pp. 7263–7271.
[12] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
[14] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "YOLOv4–5D: An effective and efficient object detector for autonomous driving," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.
[15] U. Nepal and H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs," *Sensors*, vol. 22, no. 2, p. 464, Jan. 2022.
[16] P. Singh, B. B. V. L. Deepak, T. Sethi, and M. D. P. Murthy, "Real-time object detection and tracking using color feature and motion," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2015, pp. 1236–1241.
[17] L. Xiao and T.-Q. Li, "Research on moving object detection and tracking," in *Proc. 7th Int. Conf. Fuzzy Syst. Knowl. Discovery*, Aug. 2010, pp. 2324–2327.
[18] H. Wang, P. Wang, and X. Qian, "MPNET: An end-to-end deep neural network for object detection in surveillance video," *IEEE Access*, vol. 6, pp. 30296–30308, 2018.
[19] Y. Liu, Y. Lu, Q. Shi, and J. Ding, "Optical flow based urban road vehicle tracking," in *Proc. 9th Int. Conf. Comput. Intell. Secur.*, Dec. 2013, pp. 391–395.
[20] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1468–1476.

[21] Y. Yoon, T. Kim, H. Lee, and J. Park, "Road-aware trajectory prediction for autonomous driving on highways," *Sensors*, vol. 20, no. 17, p. 4703, 2020.

[22] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using LSTMs with spatial–temporal attention mechanisms," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 2, pp. 197–208, Mar. 2021.

[23] L. H. Pham, T. T. Duong, H. M. Tran, and S. V.-U. Ha, "Vision-based approach for urban vehicle detection & classification," in *Proc. 3rd World Congr. Inf. Commun. Technol. (WICT)*, Dec. 2013, pp. 305–310.

[24] L.-W. Tsai, J.-W. Hsieh, and K.-C. Fan, "Vehicle detection using normalized color and edge map," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 850–864, Mar. 2007.

[25] A. Faro, D. Giordano, and C. Spampinato, "Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1398–1412, Dec. 2011.

[26] K. Park, D. Lee, and Y. Park, "Video-based detection of street-parking violation," in *Proc. Int. Conf. Image Process., Comput. Vis., Pattern Recognit. (IPCV)*, 2007, pp. 152–156.

[27] P. Negri, X. Clady, S. M. Hanif, and L. Prevost, "A cascade of boosted generative and discriminative classifiers for vehicle detection," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, pp. 1–12, Dec. 2008.

[28] Indrabayu, R. Y. Bakti, I. S. Areni, and A. A. Prayogi, "Vehicle detection and tracking using Gaussian mixture model and Kalman filter," in *Proc. Int. Conf. Comput. Intell. Cybern.*, 2016, pp. 115–119.

[29] V. Rin and C. Nuthong, "Front moving vehicle detection and tracking with Kalman filter," in *Proc. IEEE 4th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Feb. 2019, pp. 304–310.

[30] Z. Chen, N. Pears, M. Freeman, and J. Austin, "Road vehicle classification using support vector machines," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, Nov. 2009, pp. 214–218.

[31] Q. B. Truong and B. R. Lee, "Vehicle detection algorithm using hypothesis generation and verification," in *Proc. Int. Conf. Intell. Comput.*, Berlin, Germany, 2009, pp. 534–543.

[32] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.

[33] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," 2016, *arXiv:1605.06409*.

[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[36] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding Yolo series in 2021," 2021, *arXiv:2107.08430*.

[37] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "YOLOv6: A single-stage object detection framework for industrial applications," 2022, *arXiv:2209.02976*.

[38] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, *arXiv:2207.02696*.

[39] W. Fang, L. Wang, and P. Ren, "Tinier-YOLO: A real-time object detection method for constrained environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2020.

[40] F. Zhang, F. Yang, C. Li, and G. Yuan, "CMNet: A connect- and-merge convolutional neural network for fast vehicle detection in urban traffic surveillance," *IEEE Access*, vol. 7, pp. 72660–72671, 2019.

[41] C.-J. Lin and J.-Y. Jhang, "Intelligent traffic-monitoring system based on YOLO and convolutional fuzzy neural networks," *IEEE Access*, vol. 10, pp. 14120–14133, 2022.

[42] S. Xie, C. Liu, J. Gao, X. Li, J. Luo, B. Fan, J. Chen, H. Pu, and Y. Peng, "Diverse receptive field network with context aggregation for fast object detection," *J. Vis. Commun. Image Represent.*, vol. 70, Jul. 2020, Art. no. 102770.

[43] Y. Xue, Y. Li, S. Liu, X. Zhang, and X. Qian, "Crowd scene analysis encounters high density and scale variation," *IEEE Trans. Image Process.*, vol. 30, pp. 2745–2757, 2021.

[44] Y. Xue, Y. Li, S. Liu, P. Wang, and X. Qian, "Oriented localization of surgical tools by location encoding," *IEEE Trans. Biomed. Eng.*, vol. 69, no. 4, pp. 1469–1480, Apr. 2021.

[45] X. Li, S. Lai, and X. Qian, "DBCFace: Towards pure convolutional neural network face detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 1792–1804, Apr. 2021.

[46] X. Wang, S. Lai, Z. Chai, X. Zhang, and X. Qian, "SPGNet: Serial and parallel group network," *IEEE Trans. Multimedia*, vol. 24, pp. 2804–2814, 2021.

[47] Y. Wu, S. Feng, X. Huang, and Z. Wu, "L4Net: An anchor-free generic object detector with attention mechanism for autonomous driving," *IET Comput. Vis.*, vol. 15, no. 1, pp. 36–46, Feb. 2021.

[48] H. R. Alsanad, O. N. Ucan, M. Ilyas, A. U. R. Khan, and O. Bayat, "Real-time fuel truck detection algorithm based on deep convolutional neural network," *IEEE Access*, vol. 8, pp. 118808–118817, 2020.

[49] P. Adarsh, P. Rathi, and M. Kumar, "YOLO v3-tiny: Object detection and recognition using one stage improved model," in *Proc. 6th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Mar. 2020, pp. 687–694.

[50] Q. Liu, X. Fan, Z. Xi, Z. Yin, and Z. Yang, "Object detection based on Yolov4-tiny and improved bidirectional feature pyramid network," *J. Phys., Conf.*, vol. 2209, no. 1, Feb. 2022, Art. no. 012023.

NAFISEH ZAREI received the B.Eng. degree in electrical and electronics engineering from the Kashan University of Technology. She is currently pursuing the Ph.D. degree with Isfahan University, Iran. Her research interests include the fields of machine vision, image processing, and deep learning.

PAYMAN MOALLEM was born in Tehran, Iran, in 1970. He received the B.Sc. degree in electronics engineering from the Isfahan University of Technology, Isfahan, Iran, in 1992, and the M.Sc. degree in electronics engineering and the Ph.D. degree in electrical engineering from the Amirkabir University of Technology, Tehran, in 1996 and 2003, respectively. From 1994 to 2002, he conducted research for the Iranian Research Organization Science and Technology on the topics, such as parallel processing, robot stereo vision, and DSP boards development. In 2003, he joined the University of Isfahan, Isfahan, as an Assistant Professor, where he was promoted to an Associate Professor and a Full Professor, in 2010 and 2015, respectively. He has authored more than 300 papers published in peer-reviewed journals and conference proceedings, and five books. His research interests include remote sensing, image processing and analysis, computer vision, neural networks, and pattern recognition.

MOHAMMADREZA SHAMS received the B.S. degree from the Isfahan University of Technology, in 2008, the M.S. degree from the University of Tehran, in 2012, and the Ph.D. degree from the University of Isfahan, Iran, in 2017, all in computer engineering. He is currently an Assistant Professor with the Computer Engineering Department, University of Isfahan (Shahreza Campus). His research interests include data mining, text mining, computer vision, and deep learning.

● ● ● ●

Appendix B: Sample Code

LOGIN

```python
import smtplib
import re
import tkinter
from tkinter import FLAT
import For_pass
import customtkinter
from PIL import ImageTk, Image
from plyer import notification
from pyrebase import pyrebase
import register
import Login


customtkinter.set_appearance_mode("System")  # Modes: system (default), light, dark
customtkinter.set_default_color_theme("green")  # Themes: blue (default), dark-blue, green


app = customtkinter.CTk()  # creating cutstom tkinter window
app.geometry("600x440")
app.title('Login')
config = {
    "apiKey": "AIzaSyCQ2jsvICouZs7m7TA27a2u0MIRiLEKFZE",
    "authDomain": "aods-668cc.firebaseapp.com",
    "database": "https://aods-668cc-default-rtdb.firebaseio.com",
    "projectId": "aods-668cc",
    "storageBucket": "aods-668cc.appspot.com",
    "messagingSenderId": "34841860662",
    "appId": "1:34841860662:web:bcfce82f0319dd1208d697",
    "measurementId": "G-SNMQQ84LS2",
    "databaseURL": "https://aods-668cc-default-rtdb.firebaseio.com",
    "serviceAccount": "aods-668cc-firebase-adminsdk-r20v3-238f70f53a.json"
}
firebase = pyrebase.initialize_app(config)
database = firebase.database()
snapshot1 = database.child("Users").get()
```

```python
def validate_pass(number):
    pattern = r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[A-Za-z\d]{8,}$'
    if re.match(pattern, number):
        return True
    else:
        return False


def open_input_dialog_event1(u):
    dialog = customtkinter.CTkInputDialog(text="Enter New Password:", title="Reset
Password",show="*")
    passw = dialog.get_input()
    if validate_pass(passw):
        open_input_dialog_event2(u,passw)
    else:
        notification.notify(
            title='AODS',
            message='Should contain 8 characters including lowercase,uppercase and digits',
            app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
            timeout=10, toast=True, ticker='AODS'
        )
        open_input_dialog_event1(u)


def open_input_dialog_event2(u,pa):
    dialog = customtkinter.CTkInputDialog(text="Confirm New Password:", title="Reset
Password",show="*")
    passe = dialog.get_input()
    if pa==passe:
        user1 = u.replace('.', ',')
        database.child("Users").child(user1).child("Password").set(passe)
        notification.notify(
            title='AODS',
            message='Password Changing Successful',
            app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
            timeout=10, toast=True, ticker='AODS'
        )
    else:
```

```python
        notification.notify(
            title='AODS',
            message='Password Doesn\'t Match',
            app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
            timeout=10, toast=True, ticker='AODS'
        )
        open_input_dialog_event1(u)


def forgot_password():
    dialog = customtkinter.CTkInputDialog(text="Enter Username:", title="Reset Password",show="")
    user = dialog.get_input()
    if user is not None:
        try:
            u = database.child("Users").get()
            user1 = user.replace('.', ',')
            if user1 in u.val():
                print("k")
                open_input_dialog_event(user)
            else:
                notification.notify(
                    title='AODS',
                    message='User not Registered Contact Admin',
                    app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
                    timeout=10, toast=True, ticker='AODS'
                )
        except Exception as e:
            print(e)
    else:
        notification.notify(
            title='AODS',
            message='Enter username to Continue',
            app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
            timeout=10, toast=True, ticker='AODS'
        )


def open_input_dialog_event(u):
```

```python
config = {
    "apiKey": "AIzaSyCQ2jsvICouZs7m7TA27a2u0MIRiLEKFZE",
    "authDomain": "aods-668cc.firebaseapp.com",
    "database": "https://aods-668cc-default-rtdb.firebaseio.com",
    "projectId": "aods-668cc",
    "storageBucket": "aods-668cc.appspot.com",
    "messagingSenderId": "34841860662",
    "appId": "1:34841860662:web:bcfce82f0319dd1208d697",
    "measurementId": "G-SNMQQ84LS2",
    "databaseURL": "https://aods-668cc-default-rtdb.firebaseio.com",
    "serviceAccount": "aods-668cc-firebase-adminsdk-r20v3-238f70f53a.json"
}
firebase = pyrebase.initialize_app(config)
database = firebase.database()
snapshot1 = database.child("Users").get()
dialog = customtkinter.CTkInputDialog(text="Enter OTP send to " + u + ":", title="Reset
Password",show="*")
o=For_pass.forpass(u)
otp = dialog.get_input()
if otp==o:
    notification.notify(
        title='AODS',
        message='Verification Complete',
        app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
        timeout=10, toast=True, ticker='AODS'
    )
    print(u)
    open_input_dialog_event1(u)

else:
    notification.notify(
        title='AODS',
        message='Verification Failed',
        app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
        timeout=10, toast=True, ticker='AODS'
    )
```

```python
def button_function():
    snapshot = database.child("Users").get()
    user = entry1.get()
    print(user)
    user=user.replace('.',',')
    passw = entry2.get()

    if user in snapshot.val():
        emily = database.child("Users").child(user).child("Password").get()
        print(user)
        i = emily.val()
        if (i == passw):
            notification.notify(
                title='AODS',
                message='Login Successful',
                app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
                timeout=10, toast=True, ticker='AODS'
            )
            app.destroy()
            Login.func4()
        else:
            print("")

            notification.notify(
                title='AODS',
                message='Invalid Login',
                app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
                timeout=10  # Duration in seconds for the notification to stay on the screen
            )
    else:
        notification.notify(
            title='AODS',
            message='Invalid Username',
            app_icon="C:\\Users\\DELL\\Downloads\\istockphoto-1085043668-612x612 (4).ico",
            timeout=10)  # Duration in seconds for the notification to stay on the screen
img1 = ImageTk.PhotoImage(Image.open(r".\images\pattern.png"))
```

```python
l1 = customtkinter.CTkLabel(master=app, image=img1)
l1.pack()

frame = customtkinter.CTkFrame(master=l1, width=320, height=360, corner_radius=15)
frame.place(relx=0.5, rely=0.5, anchor=tkinter.CENTER)

l2 = customtkinter.CTkLabel(master=frame, text="Log into AODS", font=('Century Gothic', 20))
l2.place(x=50, y=45)

entry1 = customtkinter.CTkEntry(master=frame, width=220, placeholder_text='Username')
entry1.place(x=50, y=110)

entry2 = customtkinter.CTkEntry(master=frame, width=220, placeholder_text='Password', show="*")
entry2.place(x=50, y=165)
button1 = customtkinter.CTkButton(master=frame, width=220, text="Login",
command=button_function, corner_radius=6)
button1.place(x=50, y=240)
def hide(entry21, hide_image1, show_image1):
    show_button = tkinter.Button(frame, image=hide_image1, command=lambda: show(entry21,
hide_image1, show_image1),
                    relief=FLAT,
                    activebackground="grey"
                    , borderwidth=0, background="#343638", cursor="hand2")
    show_button.place(x=310, y=212)
    entry21.configure(show='*')
def show(entry21, hide_image1, show_image1):
    hide_button = tkinter.Button(frame, image=show_image1, command=lambda: hide(entry21,
hide_image1, show_image1),
                    relief=FLAT,
                    activebackground="white"
                    , borderwidth=0, background="#343638", cursor="hand2")
    hide_button.place(x=310, y=212)
    entry21.configure(show='')
hide_image = ImageTk.PhotoImage(Image.open(r".\images\hide2.png"))
show_image = ImageTk.PhotoImage(Image.open(r".\images\show1.png"))
```

```python
show_button = tkinter.Button(frame, image=hide_image, command=lambda: show(entry2,
hide_image, show_image),
                relief=FLAT,
                activebackground="grey"
                , borderwidth=0, background="#343638", cursor="hand2")
show_button.place(x=310, y=212)
l3 = tkinter.Button(
    frame,
    text="Forgot Password",
    fg="#206DB4",
    font=("yu gothic ui Bold", 12),
    bg="#2B2B2B",
    bd=0,
    activebackground="#272A37",
    activeforeground="#ffffff",
    cursor="hand2",
    command=lambda: forgot_password(),
)
l3.place(x=210, y=250)
app.mainloop()
```

VIOLATORS LIST

```python
import _thread
import concurrent.futures
import multiprocessing
import threading
import tkinter
from threading import Thread
import numpy as np
import PIL
import customtkinter
import tkinter as tk
from CTkTable import CTkTable
import RTO
from pyrebase import pyrebase
```

```python
import tkinter.messagebox
import reg_fun
import ray
import main
import imageio
from PIL import ImageTk
from PIL import Image


config = {
    "apiKey": "AIzaSyCQ2jsvICouZs7m7TA27a2u0MIRiLEKFZE",
    "authDomain": "aods-668cc.firebaseapp.com",
    "database": "https://aods-668cc-default-rtdb.firebaseio.com",
    "projectId": "aods-668cc",
    "storageBucket": "aods-668cc.appspot.com",
    "messagingSenderId": "34841860662",
    "appId": "1:34841860662:web:bcfce82f0319dd1208d697",
    "measurementId": "G-SNMQQ84LS2",
    "databaseURL": "https://aods-668cc-default-rtdb.firebaseio.com",
    "serviceAccount": "aods-668cc-firebase-adminsdk-r20v3-238f70f53a.json"
}

try:
    firebase = pyrebase.initialize_app(config)
    database = firebase.database()
    s1 = database.child("Users").get()
except Exception as e:
    print(e)
class App(customtkinter.CTk):
    def __init__(self):
        super().__init__()
        self.tab = None
        self.table = None
        self.stop_flag = threading.Event()
        self.value = [["SL.No", "Registration Number", "Picture"]]
        customtkinter.set_default_color_theme("blue")
```

```python
video_path = "video1.mp4"
self.title("AODS-MVD")
wid = self.winfo_screenwidth()
hei = self.winfo_screenheight()
self.geometry("{}x{}+0+0".format(wid, hei))
self.state('zoomed')
self.resizable(0,1)
self.grid_columnconfigure(1, weight=1)
self.grid_columnconfigure((2, 3), weight=0)
self.grid_rowconfigure((0, 1, 2), weight=1)
self.sidebar_frame = customtkinter.CTkFrame(self, width=180, height=hei, corner_radius=0)
self.sidebar_frame.grid(row=0, column=0, sticky='nsew')
self.sidebar_frame.grid_rowconfigure(4, weight=1)
self.logo_label = customtkinter.CTkLabel(self.sidebar_frame, text="AODS",
                          font=customtkinter.CTkFont(size=20, weight="bold"))
self.logo_label.grid(row=0, column=0, padx=20, pady=(20, 10))
self.sidebar_button_1 = customtkinter.CTkButton(self.sidebar_frame, text="Live Cam",
                              command=self.disable_but, width=200, height=25,
                              font=customtkinter.CTkFont(size=17))
self.sidebar_button_1.grid(row=1, column=0, padx=20, pady=10)
self.sidebar_button_2 = customtkinter.CTkButton(self.sidebar_frame, text="New User",
                              command=self.sidebar_button_event, width=200, height=25,
                              font=customtkinter.CTkFont(size=17))
self.sidebar_button_2.grid(row=2, column=0, padx=20, pady=10)
self.sidebar_button_3 = customtkinter.CTkButton(self.sidebar_frame, text="Violators",
                              command=self.thread1, width=200,
                              height=25,
                              font=customtkinter.CTkFont(size=17))
self.sidebar_button_3.grid(row=3, column=0, padx=20, pady=10)
self.string_input_button = customtkinter.CTkButton(self.sidebar_frame, text="No. of Violations", width=200,
                                  height=25,
                                  font=customtkinter.CTkFont(size=17),
                                  command=self.open_input_dialog_event)
self.string_input_button.grid(row=4, column=0, padx=20, pady=10)
self.string_input_button.place(x=20, y=210)
```

```python
        self.appearance_mode_label = customtkinter.CTkLabel(self.sidebar_frame, text="Appearance
Mode:", anchor="w",
                                              font=customtkinter.CTkFont(weight="bold"))
        self.appearance_mode_label.grid(row=5, column=0, padx=20, pady=(10, 0))
        self.appearance_mode_optionmenu = customtkinter.CTkOptionMenu(self.sidebar_frame,
                                              values=["Dark", "Light", "System"],
                                              command=self.change_appearance_mode_event,
                                              width=150)
        self.appearance_mode_optionmenu.grid(row=6, column=0, padx=20, pady=(10, 10))
        self.scaling_label = customtkinter.CTkLabel(self.sidebar_frame, text="Scaling:", anchor="w",
                                    font=customtkinter.CTkFont(weight="bold"))
        self.scaling_label.grid(row=7, column=0, padx=20, pady=(10, 0))
        self.scaling_optionmenu = customtkinter.CTkOptionMenu(self.sidebar_frame,
                                    values=["100%", "90%", "80%", "110%", "120%"],
                                    command=self.change_scaling_event, width=150)
        self.scaling_optionmenu.grid(row=8, column=0, padx=20, pady=(10, 20))
        self.appearance_mode_optionmenu.set("Dark")
        self.scaling_optionmenu.set("100%")
        self.slider_progressbar_frame = customtkinter.CTkFrame(self, width=100)
        self.slider_progressbar_frame.place(relx=0.5, rely=0.5, anchor=tk.CENTER)
        self.slider_progressbar_frame.grid(row=0, column=1, padx=(20, 0), pady=(20, 0),
sticky="nsew")
        self.slider_progressbar_frame.grid_columnconfigure(0, weight=1)
        self.slider_progressbar_frame.grid_rowconfigure(4, weight=1)
        self.progressbar_1 = customtkinter.CTkProgressBar(self.slider_progressbar_frame,
width=1200)
        self.progressbar_1.grid(row=1, column=0, padx=(10, 10), pady=(5, 10), sticky="ew")
        self.progressbar_1.configure(mode="indeterminate")
        self.progressbar_1.start()
        self.progressbar_2 = customtkinter.CTkProgressBar(self.slider_progressbar_frame,
width=1200)
        self.progressbar_2.grid(row=3, column=0, padx=(10, 10), pady=(730, 10), sticky="ew")
        self.progressbar_2.configure(mode="indeterminate")
        self.progressbar_2.start()
        self.canvas1 = tk.Canvas(self, width=800, height=800, bg='#2B2B2B')
        self.canvas1.grid(row=0, column=1, padx=(35, 12), pady=(50, 30), sticky="nesw")
```

```python
        self.canvas1.grid_columnconfigure(0, weight=1)
        self.canvas1.grid_rowconfigure(0, weight=1)
        self.sidebar_frame.grid_rowconfigure(4, weight=1)
        self.video_path = "video1.mp4"
        self.video = imageio.get_reader(self.video_path)
        self.current_frame = None
        # self.user.place(relx=0.5, rely=0.5, anchor=tk.CENTER)
        # tk.Misc.lift(self.canvas1)
        # tk.Misc.lift(self.sidebar_frame)
        self.slider_progressbar_frame.rowconfigure(0, weight=1)
        self.slider_progressbar_frame.columnconfigure(0, weight=1)
        self.l1 = None
        self.t1 = None
        self.img = None
        self.s = None
        self.tk1 = None
        self.flag = 0
        self.flag1 = 0
        self.check_something()
    def thread1(self):
        threading.Thread(target=self.start1).start()
        self.sidebar_button_3.configure(state='disabled')
        self.sidebar_button_2.configure(state="disabled")
        self.sidebar_button_1.configure(state="disabled")
    def check_something(self):
        print("check0: " + threading.current_thread().getName())
        for thread in threading.enumerate():
            print(f"Thread name: {thread.name}")
        if self.flag == 1:
            self.sidebar_button_event1()
        self.after(1000, self.check_something)

    def disable_but(self):
        self.sidebar_button_1.configure(state="disabled")
        self.sidebar_button_3.configure(state='normal')
        if self.t1 is not None:
```

```python
            self.t1.destroy()
            self.t1 = None
        if self.l1 is not None:
            if self.l1.winfo_exists():
                self.l1.destroy()
                self.l1 = None
        if self.current_frame is None:
            self.update_frame()


    def play_video(self):
        for frame in self.video.iter_data():
            image = Image.fromarray(frame)
            self.current_frame = ImageTk.PhotoImage(image)
            self.canvas1.create_image(0, 0, image=self.current_frame, anchor=tkinter.NW)
            self.canvas1.place(relx=0.5, rely=0.5, anchor=tk.CENTER)


    def update_frame(self):
        if self.current_frame is not None:
            self.canvas1.delete(self.current_frame)


        try:
            frame = self.video.get_next_data()
            image = Image.fromarray(frame)
            self.current_frame = ImageTk.PhotoImage(image)
            self.canvas1.create_image(0, 0, image=self.current_frame, anchor=tkinter.NW)
        except imageio.core.format.CannotReadFrameError:
            return


        self.after(50, self.update_frame)
    def open_input_dialog_event(self):
        dialog = customtkinter.CTkInputDialog(text="Registration Number:", title="Violations")
        no = dialog.get_input()
        n = RTO.rto(no, database)
        print(n)
        if n is None:
            tkinter.messagebox.showinfo("AODS", "No History Of Violations Found")
```

```python
        else:
            tkinter.messagebox.showinfo("AODS",
                                "The vehicle registered " + no + " has violated the rules " + n + " times")
    def change_appearance_mode_event(self, new_appearance_mode: str):
        customtkinter.set_appearance_mode(new_appearance_mode)


    def change_scaling_event(self, new_scaling: str):
        new_scaling_float = int(new_scaling.replace("%", "")) / 100
        customtkinter.set_widget_scaling(new_scaling_float)
    def new_user(self, user, user1, user2, email, mobile, passw, passc):
        print("clicg")
        u = user.get()
        u1 = user1.get()
        u2 = user2.get()
        e = email.get()
        m = mobile.get()
        p = passw.get()
        pc = passc.get()
        t = reg_fun.reg_func(u, u1, u2, e, m, p, pc, database)
        if t:
            self.l1.destroy()
            self.l1 = None
    def loading(self):
        print("loading: " + threading.current_thread().getName())
        tk1 = customtkinter.CTk()
        tk1.title("Collecting Information")
        tk1.eval('tk::PlaceWindow . center')
        tk1.geometry("320x75")
        labl=customtkinter.CTkLabel(tk1,text="Please Wait...\nDo not close this window...")
        labl.grid(row=0,column=0)
        prog = customtkinter.CTkProgressBar(tk1, width=300)
        prog.grid(row=1, column=0, padx=(10, 10), pady=(5, 10), sticky="ew")
        prog.configure(mode="indeterminate")
        prog.start()
        def stop():
            tk1.destroy()
```

```python
    def check_something1():
        # print(threading.current_thread().getName())
        x=-1
        print("checksomething1: " + threading.current_thread().getName())
        if self.flag1 == 1:
            self.flag1 = 0
            x = 0
            print(self.flag1)
            # self.tk1.destroy()
            stop()
            self.sidebar_button_2.configure(state="normal")
            self.sidebar_button_1.configure(state="normal")
            for thread in threading.enumerate():
                print(f"Thread name: {thread.name}")
        elif x==-1:
            tk1.after(1000, check_something1)
    check_something1()
    tk1.mainloop()
def start1(self):
    print("start1: "+threading.current_thread().getName())
    def start():
        # Define the target function for the new thread
        print("start: " + threading.current_thread().getName())
        self.flag = 0
        print("hi")
        self.s = database.child("Violators").get()
        print(len(self.s.each()))
        k = 1
        self.value = [["SL.No", "Registration Number", "Picture"]]
        self.img = []
        for j in range(1,len(self.s.each())):
            print(j)
            num=database.child("Plate").child(j).get()
            self.value.append([j, num.val(), 'h'])
            print(self.value)
            k = k + 1
```

```python
        for j in range(1,len(self.s.each())):
            emily = database.child("Violators").child(j).child("Pic").get()
            a = []
            for i in emily.each():
                a.append(i.val())
            img = PIL.Image.fromarray(np.uint8(a))
            img = img.resize((300,268))
            img1 = ImageTk.PhotoImage(img)
            self.img.append(img1)
        self.flag = 1
    thr1 = threading.Thread(target=start)
    thr1.start()
    thr = threading.Thread(target=self.loading)
    thr.start()
    for thread in threading.enumerate():
        print(f"Thread name: {thread.name}")
    return
def sidebar_button_event1(self):
    self.flag = 0
    self.sidebar_button_1.configure(state="normal")
    if self.l1 is not None:
        if self.l1.winfo_exists():
            self.l1.destroy()
            self.l1 = None
    if self.t1 is None:
        self.t1 = customtkinter.CTkLabel(master=self.canvas1, text="")
        self.t1.pack()
        self.tab = customtkinter.CTkScrollableFrame(master=self.t1, width=1500, height=800)
        self.tab.grid(row=0, column=0)
        self.table = CTkTable(master=self.tab, column=3, values=self.value)
        self.tab.grid_rowconfigure(0, weight=1)
        self.tab.grid_columnconfigure(0, weight=1)
        self.table.pack(expand=True, fill="both", padx=20, pady=20)
        labels = {}
        try:
            self.table.configure(values=self.value)
```

```python
        except Exception as e:
            print(e)
        k = 1
        j = 0
        print("for")
        for i in range(len(self.s.each())-1):
            print(i)
            p = f"Label {i + 1}"
            if k % 2 == 0:
                fg = "#242424"
            else:
                fg = "#333333"
            img1 = self.img[i]
            lab = customtkinter.CTkLabel(master=self.table, text="", image=img1, width=300,
height=300,
                              fg_color=fg)
            lab.grid(row=k, column=2, sticky='nsew')
            k = k + 1
            lab.lift()
            labels[p] = lab
        # threads=[]
        for thread in threading.enumerate():
            print(f"Thread name: {thread.name}"
        self.flag1=1
    def sidebar_button_event(self):
        print("sidebar_button click")
        if self.t1 is not None:
            if self.t1.winfo_exists():
                self.t1.destroy()
                self.t1 = None
        if self.l1 is not None:
            if self.l1.winfo_exists():
                print("")
        else:
            self.sidebar_button_3.configure(state='normal')
            self.sidebar_button_1.configure(state="normal")
```

```python
img1 = ImageTk.PhotoImage(Image.open(r".\images\pattern.png"))
self.l1 = customtkinter.CTkLabel(master=self.canvas1, image=img1, text="")
self.l1.pack()
self.user = customtkinter.CTkFrame(master=self.l1, width=320, height=400, corner_radius=6)
self.user.place(relx=0.5, rely=0.5, anchor=tkinter.CENTER)
l2 = customtkinter.CTkLabel(master=self.user, text="Sign In AODS", text_color='green',
                    font=('Century Gothic', 20))
l2.place(x=50, y=20)
self.entry0 = customtkinter.CTkEntry(master=self.user, width=220, height=30,
placeholder_text='Name')
self.entry0.place(x=50, y=68)
self.entry1 = customtkinter.CTkEntry(master=self.user, width=111, height=30,
                        placeholder_text='M.Name(can skip)')
self.entry1.place(x=50, y=100)
self.entry01 = customtkinter.CTkEntry(master=self.user, width=108, height=30,
placeholder_text='Last Name')
self.entry01.place(x=162, y=100)
self.entry2 = customtkinter.CTkEntry(master=self.user, width=220, height=30,
placeholder_text='Email')
self.entry2.place(x=50, y=150)
self.entry3 = customtkinter.CTkEntry(master=self.user, width=220, height=30,
                        placeholder_text='Mobile Number')
self.entry3.place(x=50, y=200)
self.entry4 = customtkinter.CTkEntry(master=self.user, width=220, height=30,
placeholder_text='Password',
                        show='*')
self.entry4.place(x=50, y=250)
self.entry5 = customtkinter.CTkEntry(master=self.user, width=220, height=30,
                        placeholder_text='Confirm Password', show='*')
self.entry5.place(x=50, y=300)
self.firstName_icon = ImageTk.PhotoImage(Image.open(r".\images\user1.png"))
firstName_icon_Label = customtkinter.CTkLabel(master=self.user, text="",
image=self.firstName_icon,
                            fg_color="#2B2B2B")
firstName_icon_Label.place(x=22, y=80)
self.email_icon = ImageTk.PhotoImage(Image.open(r".\images\emailicon1.png"))
```

```python
        emain_Label = customtkinter.CTkLabel(master=self.user, text="", image=self.email_icon,
                        fg_color="#2B2B2B")
        emain_Label.place(x=22, y=150)
        self.phone_icon = ImageTk.PhotoImage(Image.open(r".\images\phoneicon1.png"))
        phone_Label = customtkinter.CTkLabel(master=self.user, text="", image=self.phone_icon,
                        fg_color="#2B2B2B")
        phone_Label.place(x=22, y=200)
        self.pass_icon = ImageTk.PhotoImage(Image.open(r".\images\passicon1.png"))
        phone_Label = customtkinter.CTkLabel(master=self.user, text="", image=self.pass_icon,
                        fg_color="#2B2B2B")
        phone_Label.place(x=22, y=250)
        self.pass_icon2 = ImageTk.PhotoImage(Image.open(r".\images\passicon2.png"))
        phone_Label2 = customtkinter.CTkLabel(master=self.user, text="", image=self.pass_icon2,
                        fg_color="#2B2B2B")
        phone_Label2.place(x=22, y=300)

        self.button1 = customtkinter.CTkButton(master=self.user, width=220, text="Login",
                            command=lambda: self.new_user(self.entry0, self.entry1,
self.entry01,

                                    self.entry2, self.entry3, self.entry4,
                                    self.entry5),
                        corner_radius=6)
        self.button1.place(x=50, y=350)
def func4():
    app = App()
    app.mainloop()
def func3():
    ray.init()
    @ray.remote
    def func1():
        app = App()
        app.mainloop()
    @ray.remote
    def func2():
        main.main()
    ray.get([func1.remote(), func2.remote()])
```

NUMBER PLATE EXTRACTION

```python
def blur(image):
    import re
    import PIL
    import cv2
    import easyocr
    import numpy
    import numpy as np
    from PIL import Image
    import page2
    from ESRGAN import test
    def thick_font(image):
        import numpy as np
        image = cv2.bitwise_not(image)
        kernel = np.ones((2, 2), np.uint8)
        image = cv2.dilate(image, kernel, iterations=1)
        image = cv2.bitwise_not(image)
        return (image)
    def thin_font(image):
        import numpy as np
        image = cv2.bitwise_not(image)
        kernel = np.ones((2, 2), np.uint8)
        image = cv2.erode(image, kernel, iterations=1)
        image = cv2.bitwise_not(image)
        return (image)
    def noise_removal(image):
        image = numpy.asarray(image)
        kernel = np.ones((1, 1), np.uint8)
        image = cv2.dilate(image, kernel, iterations=1)
        kernel = np.ones((1, 1), np.uint8)
        image = cv2.erode(image, kernel, iterations=1)
        image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
        image = cv2.medianBlur(image, 3)
        return (image)
```

```python
        sharpened = test.enhance(image)
        sharpened = test.enhance(sharpened)
        gray_img = cv2.cvtColor(sharpened, cv2.COLOR_BGR2GRAY)
        gray_img = noise_removal(gray_img)
        im_bw = thick_font(gray_img)
        sharpened = PIL.Image.fromarray(np.uint8(gray_img))
        BW = noise_removal(im_bw)
        BW = PIL.Image.fromarray(np.uint8(BW))
        BW.save("BW_image.jpg")
        image = cv2.imread(r'BW_image.jpg')
        reader1 = easyocr.Reader(['th'], gpu=False)
        ocr_result1 = reader1.readtext(image)
        reader2 = easyocr.Reader(['en'], gpu=False)
        ocr_result2 = reader2.readtext(image)
        print(ocr_result1)
        print(ocr_result2)
        print(len(ocr_result2))
        msg = "None"
        if (len(ocr_result2)) > 0:
            eng1 = ((ocr_result2[0])[1]).replace(" ", "")
            thi = ((ocr_result1[0])[1]).replace(" ", "")
            msg = ""
            if len(ocr_result2) > 2:
                for i in ocr_result1:
                    pattern = '\d{2}-\d{4}'
                    if re.match(pattern, i[1]):
                        print(i[1])
                        return(i[1])
            else:
                l = None
                j = len(eng1) - 4
                if len(eng1) > len(thi):
                    l = len(eng1)
                else:
```

```python
        l = len(thi)
        print(len((ocr_result1[0])[1]))
        print(j)
        if l > 6:
            for i in range(l):
                if i == 0:
                    if eng1[i].isnumeric():
                        msg = msg + eng1[i]
                    else:
                        msg = msg + thi[i]
                elif i > 2:
                    msg = msg + eng1[j]
                    j = j + 1
                else:
                    msg = msg + thi[i]
        else:
            msg = thi
    print(msg)
    return msg
```

VEHICLE DETECTION

```python
import threading
from time import sleep
import os
import numpy
from pyrebase import pyrebase
from ultralytics import YOLO
import cv2
import cvzone
import math
from sort import *
from datetime import datetime
import page2
import tensorflow as tf
import _thread
```

```python
def call1(id,crop):
    thr=threading.Thread(target=page2.no_reading,args=(id,crop))
    thr.start()
    thr.join()
def main():
    cap = cv2.VideoCapture("video1.mp4")  # For Video
    model = YOLO("/Yolo-Weights/yolov8l.pt")
    config = {
        "apiKey": "AIzaSyCQ2jsvICouZs7m7TA27a2u0MIRiLEKFZE",
        "authDomain": "aods-668cc.firebaseapp.com",
        "database": "https://aods-668cc-default-rtdb.firebaseio.com",
        "projectId": "aods-668cc",
        "storageBucket": "aods-668cc.appspot.com",
        "messagingSenderId": "34841860662",
        "appId": "1:34841860662:web:bcfce82f0319dd1208d697",
        "measurementId": "G-SNMQQ84LS2",
        "databaseURL": "https://aods-668cc-default-rtdb.firebaseio.com",
        "serviceAccount": "aods-668cc-firebase-adminsdk-r20v3-238f70f53a.json"
    }
    def rgb(event, x, y, flags, param):
        if event == cv2.EVENT_MOUSEMOVE:
            colorsBGR = [x, y]
            print(colorsBGR)
    cv2.namedWindow('Image')
    cv2.setMouseCallback('Image', rgb)
    def imgwrite(img):
        now = datetime.now()
        current_time = now.strftime("%d_%m_%Y_%H_%M_%S")
        filename = '%s.png' % current_time

cv2.imwrite(os.path.join(r"C:\Users\Harikrishnan\PycharmProjects\mini_project\pythonProject\img",
filename), img)
    classNames = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck", "boat",
            "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat",
            "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", "backpack",
"umbrella",
```

```python
                "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", "sports ball", "kite", "baseball
bat",
                "baseball glove", "skateboard", "surfboard", "tennis racket", "bottle", "wine glass", "cup",
                "fork", "knife", "spoon", "bowl", "banana", "apple", "sandwich", "orange", "broccoli",
                "carrot", "hot dog", "pizza", "donut", "cake", "chair", "sofa", "pottedplant", "bed",
                "diningtable", "toilet", "tvmonitor", "laptop", "mouse", "remote", "keyboard", "cell phone",
                "microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors",
                "teddy bear", "hair drier", "toothbrush"
                ]
    mask = cv2.imread("mask1.png")
    tracker = Sort(max_age=20, min_hits=3, iou_threshold=0.3)
    limits = [320, 650, 1373, 650]
    totalCount = []
    area = [(580, 831), (630, 830), (1000, 65), (978, 65)]
    area1 = [(1098, 829), (1177, 831), (1147, 89), (1126, 89)]
    area_c = []
    while True:
        success, img = cap.read()
        if not success:
            video = cv2.VideoCapture("video1.mp4")
            continue
        imgRegion = cv2.bitwise_and(img, mask)
        results = model(imgRegion, stream=True)
        detections = np.empty((0, 5))
        for r in results:
            boxes = r.boxes
            for box in boxes:
                x1, y1, x2, y2 = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                w, h = x2 - x1, y2 - y1
                conf = math.ceil((box.conf[0] * 100)) / 100
                cls = int(box.cls[0])
                currentClass = classNames[cls]
                if currentClass == "car" or currentClass == "truck" or currentClass == "bus" \
                        or currentClass == "motorbike" and conf > 0.3:
                    currentArray = np.array([x1, y1, x2, y2, conf])
```

```python
            detections = np.vstack((detections, currentArray))
    resultsTracker = tracker.update(detections)
    for result in resultsTracker:
        x1, y1, x2, y2, id = result
        x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
        w, h = x2 - x1, y2 - y1
        cx, cy = x1 + w // 2, y1 + h // 2
        results = cv2.pointPolygonTest(np.array(area1, np.int32), ((cx, cy)), False)
        results1 = cv2.pointPolygonTest(np.array(area, np.int32), ((cx, cy)), False)
        cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255), 3)
        m1 = 0
        if results >= 0 or results1 >= 0:
            crop = img[y1:y2, x1:x2]
            for i1 in area_c:
                if id == i1:
                    m1 = 1
            if m1 == 0:
                area_c.append(id)
        w, h = x2 - x1, y2 - y1
        cvzone.cornerRect(img, (x1, y1, w, h), l=9, rt=2, colorR=(255, 0, 255))
        cvzone.putTextRect(img, f' {int(id)}', (max(0, x1), max(35, y1)),
                    scale=2, thickness=3, offset=10)
        cx, cy = x1 + w // 2, y1 + h // 2
        cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
        if limits[0] < cx < limits[2] and limits[1] - 15 < cy < limits[1] + 15:
            if id in area_c:
                area_c.remove(id)
                crop = img[y1:y2, x1:x2]
                imgwrite(crop)
                dat = str(int(id))
                parent_node = 'Violators'
                data = {"Pic": crop.tolist()}
                print(type(crop))
                call1(id,crop)
                cv2.line(img, (limits[0], limits[1]), (limits[2], limits[3]), (0, 255, 0), 5)
    cv2.polylines(img, [np.array(area, np.int32)], True, (0, 255, 0), 3)
```

```python
cv2.polylines(img, [np.array(area1, np.int32)], True, (255, 0, 0), 3)
cv2.line(img, (limits[0], limits[1]), (limits[2], limits[3]), (0, 0, 255), 5)
cv2.imshow("Image", img)
cv2.waitKey(1)
```

NUMBER PLATE DETECTION

```python
import os
import tensorflow as tf
from IPython import get_ipython
from object_detection.utils import config_util
import cv2
import numpy as np
from matplotlib import pyplot as plt
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
import easyocr
from pyrebase import pyrebase
import keras_ocr
import blur
def no_reading(id, image_np):
    CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
    PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
    PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320
x320_coco17_tpu-8.tar.gz'
    TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
    LABEL_MAP_NAME = 'label_map.pbtxt'
    print(id)
    paths = {
        'WORKSPACE_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace'),
        'SCRIPTS_PATH': os.path.join('ANPR', 'Tensorflow', 'scripts'),
        'APIMODEL_PATH': os.path.join('ANPR', 'Tensorflow', 'models'),
```

```python
    'ANNOTATION_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'annotations'),
    'IMAGE_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'images'),
    'MODEL_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'models'),
    'PRETRAINED_MODEL_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace',
'pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'models',
CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'models',
CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'models',
CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH': os.path.join('ANPR', 'Tensorflow', 'workspace', 'models',
CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH': os.path.join('ANPR', 'Tensorflow', 'protoc')
  }
  files = {
    'PIPELINE_CONFIG': os.path.join('ANPR', 'Tensorflow', 'workspace', 'models',
CUSTOM_MODEL_NAME,
                        'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'],
TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
  }
  try:
    tf.config.set_visible_devices([], 'GPU')
    visible_devices = tf.config.get_visible_devices()
    for device in visible_devices:
      assert device.device_type != 'GPU'
  except:
    pass
  try:
    configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
    detection_model = model_builder.build(model_config=configs['model'], is_training=False)
    ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
    ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-11')).expect_partial()
    @tf.function
```

```python
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)
num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
label_id_offset = 1
image_np_with_detections = image_np.copy()
viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'],
    detections['detection_classes'] + label_id_offset,
    detections['detection_scores'],
    category_index,
    use_normalized_coordinates=True,
    max_boxes_to_draw=5,
    min_score_thresh=.6,
    agnostic_mode=False)
detection_threshold = 0.6
image = image_np_with_detections
scores = list(filter(lambda x: x >= detection_threshold, detections['detection_scores']))
boxes = detections['detection_boxes'][:len(scores)]
classes = detections['detection_classes'][:len(scores)]
width = image.shape[1]
height = image.shape[0]
print(boxes)
eng = ""
msg = "None"
for idx, box in enumerate(boxes):
```

```python
        roi = box * [height, width, height, width]
        region = image[int(roi[0]):int(roi[2]), int(roi[1]):int(roi[3])]
        msg = blur.blur(region)
        config = {
            "apiKey": "AIzaSyCQ2jsvlCouZs7m7TA27a2u0MIRiLEKFZE",
            "authDomain": "aods-668cc.firebaseapp.com",
            "database": "https://aods-668cc-default-rtdb.firebaseio.com",
            "projectId": "aods-668cc",
            "storageBucket": "aods-668cc.appspot.com",
            "messagingSenderId": "34841860662",
            "appId": "1:34841860662:web:bcfce82f0319dd1208d697",
            "measurementId": "G-SNMQQ84LS2",
            "databaseURL": "https://aods-668cc-default-rtdb.firebaseio.com",
            "serviceAccount": "aods-668cc-firebase-adminsdk-r20v3-238f70f53a.json"
        }
        firebase = pyrebase.initialize_app(config)
        database = firebase.database()
        len1 = database.child("Violators").get()
        print(len(len1.each()))
        print(type(len(len1.each())))
        len2 = len(len1.each())
        database.child("Violators").child(len2).child("Pic").set(image_np.tolist())
        database.child("Plate").child(len2).set(msg)
        len3 = database.child("RTO").get()
        fla = 0
        for k in len3.val():
            if k == msg:
                num = database.child("RTO").child(msg).child("num").get()
                num1 = int(num.val())
                num1 = num1 + 1
                database.child("RTO").child(msg).child("num").set(str(num1))
                fla = 1
                break
        if fla == 0:
            database.child("RTO").child(msg).child("num").set("1")
except Exception as e:
```

```
        print(e)
```

IMAGE ENHANCEMENT

```python
import os.path as osp
import glob
import cv2
import numpy as np
import torch
from ESRGAN import RRDBNet_arch as arch

def enhance(img):
    model_path = 'ESRGAN/models/RRDB_ESRGAN_x4.pth'  # models/RRDB_ESRGAN_x4.pth OR
models/RRDB_PSNR_x4.pth
    device = torch.device('cuda')  # if you want to run on CPU, change 'cuda' -> cpu
    # device = torch.device('cpu')

    test_img_folder = 'LR/*'

    model = arch.RRDBNet(3, 3, 64, 23, gc=32)
    model.load_state_dict(torch.load(model_path), strict=True)
    model.eval()
    model = model.to(device)

    print('Model path {:s}. \nTesting...'.format(model_path))
    # base = osp.splitext(osp.basename(img))[0]
    img = img * 1.0 / 255
    img = torch.from_numpy(np.transpose(img[:, :, [2, 1, 0]], (2, 0, 1))).float()
    img_LR = img.unsqueeze(0)
    img_LR = img_LR.to(device)

    with torch.no_grad():
        output = model(img_LR).data.squeeze().float().cpu().clamp_(0, 1).numpy()
    output = np.transpose(output[[2, 1, 0], :, :], (1, 2, 0))
    output = (output * 255.0).round()
    # cv2.imwrite('ESRGAN/results/{:s}_rlt.png'.format(base), output)
    cv2.imwrite('ESRGAN/results/o_rlt.png', output)
```

```python
output=cv2.imread('ESRGAN/results/o_rlt.png')
return output
```

Appendix C: CO-PO And CO-PSO Mapping

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS6 22T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS6 22T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS6 22T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS6 22T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS6 22T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS6 22T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS6 22T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS6 22T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS6 22T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS6 22T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS6 22T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS6 22T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS6 22T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS6 22T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS6 22T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS6 22T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS6 22T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS6 22T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS6 22T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS6 22T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS6 22T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS6 22T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS6 22T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS6 22T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS6 22T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS6 22T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS6 22T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS6 22T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS6 22T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS6 22T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS6 22T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS6 22T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS6 22T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS6 22T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS6 22T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS6 22T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS6 22T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS6 22T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS6 22T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS6 22T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS6 22T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS6 22T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS6 22T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS6 22T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS6 22T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS6 22T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS6 22T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS6 22T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS6 22T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS6 22T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS6 22T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS6 22T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS6 22T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS6 22T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS6 22T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS6 22T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS6 22T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS6 22T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS6 22T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS6 22T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.