Detecting and Counting different Vehicles in the real time Traffic Signal using deep learning

MSc Research Project
Programme Name

Stephen Angelo Kumar
Student ID: x21220123

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Stephen Angelo Kumar
Student ID:	x21220123
Programme:	MSC Cloud Computing
Year:	2022
Module:	Research Project
Supervisor:	Diego Lugones
Submission Due Date:	18/09/23
Project Title:	Detecting and Counting different Vehicles in the realtime Traffic Signal using deep learning
Word Count:	5892
Page Count:	22

Signature:	STEPHEN ANGELO KUMAR
Date:	18th September 2023

Detecting and Counting different Vehicles in the real time Traffic Signal using deep learning

Stephen Angelo Kumar
x21220123

Abstract

Traffic congestion is a pressing global issue. This research proposes an ensemble approach to improve traffic management using traffic cameras, real-time video analysis, time series forecasting, and rules derived from traffic patterns. The project develops an algorithm to control green light duration and ensure efficient traffic flow. The ensemble approach optimizes green time using deep learning techniques for object detection and time series forecasting to predict traffic volumes. For real-time object detection, YOLOv4 achieved the highest mean Average Precision at 42.44%. The ensemble approach integrates video analysis and time series projections to dynamically manage traffic. This study demonstrates the potential of combining computer vision and predictive analytics to optimize traffic signals and reduce congestion.

1 Introduction

The widespread use of automobiles has revolutionized daily commutes. However, it also raises concerns about traffic congestion and its environmental impact. As urban centers grow, congestion worsens. This has prompted the development of intelligent traffic management systems. These intelligent systems monitor green light timings and analyze traffic data. Their aim is to reduce congestion in high-traffic areas. Connected Traffic Management Systems (CTMS) can significantly improve mobility and reduce environmental impact.

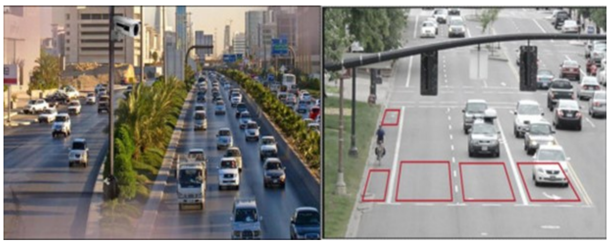


Figure 1: Operation of the Connected and Automated Traffic System (3)

This essay explores the potential of emerging technologies like CNNs and surveillance cameras for traffic management. It focuses on real-time video feeds in Bhubaneswar, India, where urbanization and population growth have increased automobile usage. The proposed intelligent system aims to alleviate congestion, improve efficiency, and prevent environmental degradation. This would benefit both present and future generations

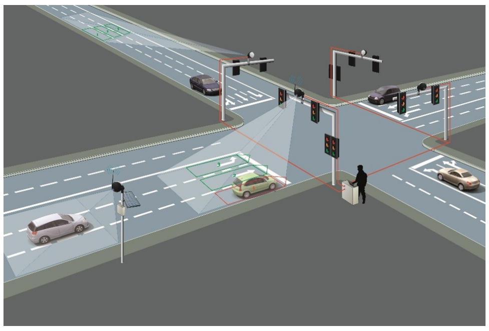


Figure 2: Automatic Traffic Detection System (Courtesy: `https://www.aldridgetrafficcontrollers.com.au/products/video-detection`)

1.0.1 Motivation

Urbanization and economic development have led to increased car usage, causing traffic congestion and pollution. Intelligent traffic management systems are needed to regulate flow and prevent congestion. This research aims to create a system using Convolutional Neural Networks and surveillance cameras to monitor traffic, identify bottlenecks, and predict future flows. The goal is to reduce commute times by 10%, fuel consumption by 5%, and carbon dioxide emissions by 7%, improving quality of life and contributing to international efforts to reduce traffic congestion and promote environmentally friendly urban transport networks.

1.0.2 Business Problem

Rapid urbanization and economic growth have led to the growth of smart cities and metro towns, causing gridlock and challenges for businesses. Delays in workplace travel reduce productivity, increase transportation costs, and cause inventory management issues. Sustainable practices face environmental impacts from traffic congestion. Intelligent Transportation Systems (ITS) can optimize traffic flow, reduce travel times, and limit environmental impact, boosting productivity, sustainability goals, and employee satisfaction. Implementing these solutions is crucial for long-term success and competitiveness in the competitive world.

1.0.3 Impact of the Solution

Traffic congestion is a significant social, economic, and environmental challenge in low-income regions. Adaptive traffic management, using AI, IoT, and big data, has been proposed as a potential solution. Implementing this strategy can reduce delays, waste time and money, and improve efficiency. By adjusting traffic patterns in real-time, it can reduce congestion, increase mobility, and potentially save money. This approach can lead to reduced congestion, increased mobility, and cost savings, ultimately benefiting society.

1.0.4 Research Hypothesis

In order to alleviate traffic congestion and improve transportation efficiency, drivers and traffic management agencies need access to reliable and timely information on traffic flow. The following concerns will serve as the basis for the study subjects that will be explored in this article.

RQ1: How can I maximise the effectiveness of the green light so that the flow of traffic is not disrupted?

RQ2: How can we determine which cars are emergency responders and get the system to act accordingly?

RQ3: Which deep learning method, when combined with computer vision techniques, performs particularly well in real time?

RQ4: How can such a large quantity of data in the form of real-time videos be processed in order to reach a conclusion regarding the optimal green light in a short length of time?

Scholars have examined potential solutions for traffic management systems in smart cities, focusing on real-time traffic prediction and connected traffic management. To address congestion, an analysis of Bhubaneswar's traffic systems and Bengaluru's challenges will be conducted. The recommended scenario's methodical approach and algorithm will be outlined, along with steps to improve system dependability.

2 Related Work

Traffic systems now integrate edge computing, cloud computing, and deep learning. Edge devices filter and analyze data locally using deep learning models. The edge sends condensed insights to the cloud for aggregation. This hybrid architecture responds faster to traffic conditions than centralized clouds alone.

2.1 Artificial Intelligence based Traffic Control Approaches Using Various Neural Network and machine learning algorithms

To improve traffic management, Alkinani et al. (2022) propose combining artificial and empirical intelligence. Their system uses the XGboost algorithm to adapt to new traffic conditions and learn from historical data in real-time. This enables detecting unusual patterns and alerting authorities, improving efficiency and safety. Simulations showed high accuracy, precision and recall, indicating scalability to different settings. By integrating AI and empirical data, this work significantly advances traffic management.

Djenouri et al. (2023) highlight Graph Convolutional Neural Networks for capturing spatial correlations in traffic data. They explain forecasting traffic flow is key for Intelligent Transportation Systems and edge computing's potential.

Popov et al. (2023) propose automated control systems and AI can better manage urban traffic. They examine systems from several angles and use neural networks to develop an intelligent traffic light manager.

Salem et al. (2022) proposed the Fusion-Based Intelligent Traffic Congestion Control System for Virtual Networks (FITCCS-VN) using machine learning and vehicular net-

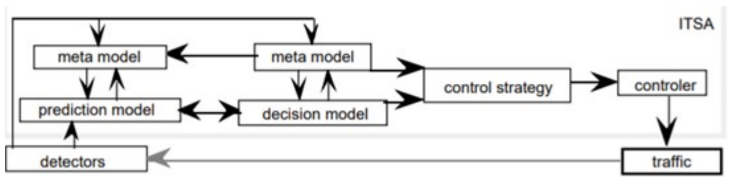


Figure 3: ITSA Controller Agent

works to monitor traffic and efficiently distribute vehicles, aiming to relieve congestion in smart cities. They found a 95% success rate, showing promise.

Haojia et al. (2022) developed a deep learning method to count vehicles in lanes from videos in real-time with up to 99% accuracy. This enables precise traffic monitoring for smart transportation.

Bilotta et al. (2022) proposed a novel convolutional bidirectional LSTM model to predict urban traffic flow 10-60 minutes ahead. Evaluated in Italy's Sii-Mobility project, it showed benefits for traffic management and sustainability.

Bhattacharya et al. (2022) used deep learning accounting for spatial and temporal correlations to forecast traffic. This accurately predicted urban patterns.

Chiang et al. (2022) built short- and long-term travel time prediction models for a Taiwanese highway using nine algorithms.

Harrou et al. (2022) developed a guided-attention hybrid deep learning model to improve pedestrian and bicycle traffic flow forecasting.

Several studies addressed traffic congestion issues prior to implementing complex smart traffic management systems (STMS). Tseng et al. (2018) focused on predicting congestion using a Support Vector Machine-based Traffic Bottleneck model to forecast slow traffic during peak hours. Allstrom et al. (2017) found advanced traveler information and management systems are key for smart city traffic initiatives to enable better management, less pollution, shorter commutes. They proposed a multi-step method for intelligent traffic light control. Du et al. (2018) used deep neural networks to predict future vehicle counts and volumes to understand traffic patterns. Bull & Thomson (2002) studied the relationship between travel time, distance, and speed. Barnes et al. (2001) developed a CNN-GRU hybrid model with attention to improve multi-modal traffic flow forecasting, enhancing robustness and flexibility. The overarching goal of these studies is providing workable solutions to urban congestion challenges using state-of-the-art algorithms and technology in traffic management.

2.2 Cloud assisted Internet of things intelligent transportation system and the traffic control system in the smart city

Extensive work has aimed to improve traffic control and safety while reducing congestion. Research has utilized computer vision to identify congestion from surveillance data.

Agent-based systems and deep learning have also been studied. Dombalyan et al. (2017) proposed a computer simulation strategy for traffic forecasting, evaluating factors affecting transportation in two Russian regions. Dzhuruk & Zedgenizov (2018) developed a novel regression-based approach to predict traffic volume, finding high volume locations negatively impact neighborhoods.

Roozemond & Danko A (1999)(20) created Intelligent Interaction Traffic Signalling Agents using AI to coordinate with infrastructure in real-time, improving efficiency. Roozemond (1996) developed the Dynamic Traffic Management and Information System combining platforms for diverse users, increasing productivity, safety, cost-effectiveness, energy efficiency, environment, and convenience. These bolster intelligent traffic management infrastructure.

Umesh Kumar et al. (2022) proposed an Adaptive Traffic-management system using Internet of Things and machine learning to detect abnormalities and adjust signals, shortening wait times, reducing congestion and accidents, and improving quality of life. Despite limitations, this contributes significantly to literature on smart transportation systems, demonstrating the potential of IoT and machine learning to enhance traffic management.

To alleviate traffic jams in smart cities, (7) proposed a Cloud-assisted Internet of Things Intelligent Transportation System (CIoT-ITS). The system utilizes IoT-connected sensors and cameras to track traffic patterns and enhance cloud-based data analysis. An algorithm calculates traffic flow to optimize signal timing. When a jam is detected, it alerts the nearest control center. Simulations showed the proposed system effectively manages vehicle flow, outperforming conventional methods. This demonstrates an effective, reliable CIoT-ITS to address smart city traffic management challenges.

Liu et al(8). introduced a cloud-assisted IoT Intelligent Transportation System to manage smart city traffic. This trustworthy, cost-effective system monitors traffic, routes it efficiently, optimizes cloud data analysis, and adapts traffic signals accordingly. The research highlights the potential of emerging technologies like cloud computing and IoT combined with intelligent algorithms to create smarter, more dynamic traffic management systems for congested cities.

2.3 Edge Computing AI-IoT Integrated Energy-efficient Intelligent Transportation System

Chen et al. (2022) reviewed literature on using AI for traffic management, noting many methods require substantial processing. They proposed edge computing as a solution to enable collaborative networks to better handle urban traffic issues by deploying responsive servers in real-time.

Zhou et al. (2022)(19) integrated a deep convolution random forest neural network into the vehicle network to increase smart city intelligence. This reduced delay costs, increased unloading utility, and achieved 98.06% recognition accuracy compared to prior methods.

Chavan et al. (2022) introduced a distributed multi-agent system using edge AI and IoT for energy-efficient smart city transportation. It aims to improve efficiency, reduce emissions, and better serve commuters and agencies. Simulations demonstrated feasibility. Wan (16) proposed an edge computing system using spatio-temporal interest points and multi-modal linear characteristics to eliminate redundant video data in intelligent transportation. This improves efficiency while reducing the cloud computing burden.

Research on intelligent traffic control and management has explored various approaches to improve efficiency and reduce congestion, including:

• Intelligent Traffic Signaling Agents using control strategies, prediction models and communication networks ((20)).
• Autonomous intelligent agents based on machine learning techniques, gaining more attention than traditional hierarchical control structures ((5)).
• Smart traffic control implementation in cities using cloud computing, wireless networks and smart grid technologies ((10)).
• Knowledge-based AI systems to provide diagnostic and control actions for traffic management ((6)).
• Short-term traffic prediction like deep neural networks to anticipate congestion (18)

These efforts demonstrate the potential of leveraging machine learning, data analysis and communication technologies for optimizing and reducing congestion in transportation networks. While several promising solutions have been explored, more work is needed to incorporate them into standard traffic control systems. Research is progressing in applying intelligent systems to traffic management, but real-world implementation and integration remains an ongoing challenge.

2.4 Findings of the above papers

The reviewed papers cover recent research on traffic management systems using cutting-edge technologies including IoT, deep learning, edge computing, and other machine learning techniques. Key motivations are reducing congestion, enabling accurate forecasting, and optimizing signals. However, there are still gaps requiring further study:

• The broader impacts of traffic on the economy, environment, and society need investigation.

• Innovations tailored for rural areas are needed.

• Developing interpretable models could improve trust and transparency.

• Exploring alternative data sources like social media could enhance systems.

• Studying impacts on vulnerable road users could improve safety.

While promising technologies are emerging for traffic management, there remain opportunities to address systemic impacts, adapt solutions for diverse environments, and ensure equity and ethical considerations are incorporated into next generation intelligent systems. A holistic, inclusive approach is key.

2.5 Research Niche

This intelligent traffic system uses cameras, real-time video analytics, and forecasting models. It monitors conditions, detects vehicles, and anticipates volume. The system adapts signal timing based on congestion to maximize flow. This proactive approach manages traffic dynamically through monitoring, prediction, and adaptive control.

3 Methodology

3.1 Research Method

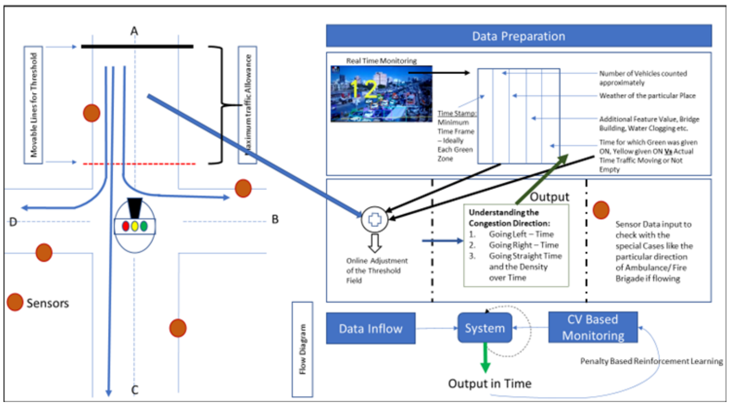

A
B
C
D
Movable Lines for Threshold
Maximum traffic Allowance
Sensors
Flow Diagram
Data Preparation
Real Time Monitoring
Time Stamp: Minimum Time Frame – Ideally Each Green Zone
Number of Vehicles counted approximately
Weather of the particular Place
Additional Feature Value, Bridge Building, Water Clogging etc.
Time for which Green was given ON, Yellow given ON Vs Actual Time Traffic Moving or Not Empty
Output
Understanding the Congestion Direction:
1. Going Left – Time
2. Going Right – Time
3. Going Straight Time and the Density over Time
Online Adjustment of the Threshold Field
Sensor Data input to check with the special Cases like the particular direction of Ambulance/ Fire Brigade if flowing
Data Inflow
System
CV Based Monitoring
Penalty Based Reinforcement Learning
Output in Time

Figure 4: System Architecture

The proposed Connected Traffic Management System architecture involves acquiring real-time data to count objects as shown in Figure 5.To enable real-time processing, the system leverages AWS EC2 instances and GPUs for accelerated video processing.

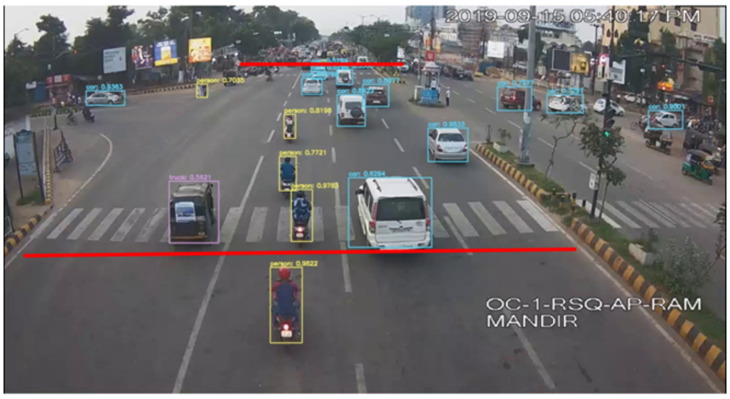


Figure 5: Region of Interest

A key step is defining the Region of Interest (ROI) for targeted object detection. After focused object detection, the next stage is preparing the dataset for optimizing green light timing. This involves detecting road density and counting vehicles present during the green light.

The system has three core components:

a) Object Detection Agent: Performs computer vision on video feeds to identify vehicle types.
b) Congestion/Density Detection Agent: Analyzes vehicle counts to detect congestion levels and hotspots.
c) Traffic Forecasting Agent: Applies algorithms to forecast traffic volumes and assign weighting.

The traffic management system involves the following key steps:

Step 1) Acquire real-time video feeds at intersections using traffic cameras.
Step 2) Perform object detection on the feeds using a customized deep learning algorithm to identify vehicle types like cars, rickshaws, bikes, buses, trucks, and pedestrians.
Step 3) Count vehicles over fixed intervals using the SORT algorithm. Formulate time series dataset with vehicle counts per lane in 10 minute slots.
Step 4) Assign unique IDs to tracked vehicles and remove IDs after vehicles leave camera view.
Step 5) Forecast short-term traffic per lane using time series algorithms. Detect density to identify cohortion.
Step 6) Ensemble forecasting and density outputs using weighted averaging and rules-based logic for adaptive signal timing.
Step 7) Retune algorithms if performance metrics not met.
Step 8) Optimize ensemble components to reduce congestion across intersection.

The system leverages video analytics, vehicle tracking, adaptive forecasting, and congestion detection to intelligently optimize traffic light timing and minimize congestion. The multi-step process allows responding in real-time to evolving traffic conditions.

4 Design & Implementation

The following object detection and tracking models will be utilized:

1. R-CNN - Performs region-based detection followed by classification. Uses selective search for region proposals to improve speed ((14)(18)).

2. Fast R-CNN - Uses a single neural network for region proposal and classification to improve performance ((14)).

3. Faster R-CNN - Uses a Region Proposal Network for faster region proposals ((14)).

4. YOLO - Frames object detection as a single regression problem for fast prediction ((11)).

5. SSD - Uses small convolutional filters to detect objects at various scales ((8)).

For vehicle counting, the SORT algorithm will be used which assigns unique IDs to tracked objects (Bewley et al., 2016).

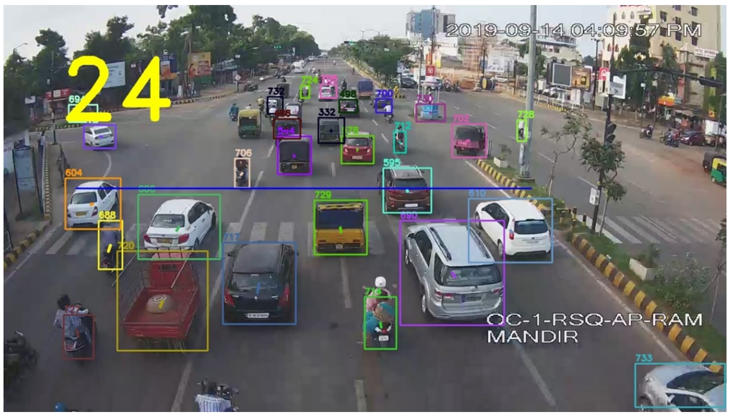


Figure 6: Region of Interest

As shown in Figure 6, these models enable real-time object detection and vehicle counting on video feeds at the intersection. The ensemble of models balances accuracy and computational efficiency for the traffic management system.

R-CNN – RCNN performs detection in a two-step process. First, it generates region proposals using selective search to focus on particular image regions. This helps improve speed compared to exhaustively searching the full image. It then extracts features from the proposed regions and classifies them using a convolutional neural network. R-CNN merges region proposals based on color spaces, textures and other attributes before classification. Overall, by combining selective region proposals with CNN-based classification, R-CNN improves accuracy and efficiency compared to previous detection approaches ((14) (18)).

Fast R-CNN – Instead of using a CNN separately on region proposals like R-CNN, Fast R-CNN passes the entire image through a single CNN to generate feature maps. It then applies selective search on the feature maps to identify regions of interest (RoIs). ROIs are pooled to fixed size vectors using RoI pooling for classification by fully connected layers ((17); (15); (13)).

Faster R-CNN – Faster R-CNN combines a region proposal network (RPN) with Fast R-CNN for improved speed and accuracy. The RPN generates region proposals directly from the feature maps, removing the selective search bottleneck. This streamlines training and increases efficiency ((12); (4)).

YOLO (You Only Look Once) is a real-time object detection algorithm that uses a single convolutional neural network to make predictions. It frames object detection as a regression problem to predict bounding boxes and class probabilities directly from full images in one evaluation. This enables fast processing speeds suitable for applications requiring high frame rates where small accuracy tradeoffs are acceptable.

YOLO is fully convolutional, composed of only convolutional layers without pooling. YOLOv3 introduces Darknet-53 for feature extraction, with 53 convolutional layers and batch normalization and Leaky ReLU activation. It avoids pooling, instead using strided convolution to downsample and retain low-level features. By eliminating the region proposal stage and taking a unified approach, YOLO can perform rapid detection with reasonable accuracy tradeoffs ((9); (2) ;(11)).

YOLO is designed to be invariant to input image size, though in practice a constant size is used. The input is a batch of images, and the output is a list of predicted bounding boxes and class probabilities. Each bounding box prediction contains the box coordinates, objectness score, and class probabilities. Using 1x1 convolutions, YOLO makes predictions at the full feature map resolution before it. In YOLOv3, each cell predicts a fixed number of bounding boxes B. These boxes may specialize in detecting certain objects. Each box is described by 5+C attributes - the center coordinates, dimensions, objectness score, and C class confidences. Rather than predict heights and widths directly, YOLOv3 predicts offsets to default bounding boxes called anchors. It uses 3 anchors per cell, predicting 3 boxes. The network output is transformed into box predictions using the anchor boxes as follows:

bbox_pred = anchor_box + exponent(output)

This scheme of predicting anchor offsets allows using predefined priors to normalize the bounding box regression task. YOLO's unified architecture enables fast prediction of class probabilities and bounding box coordinates directly from full images in a single pass.

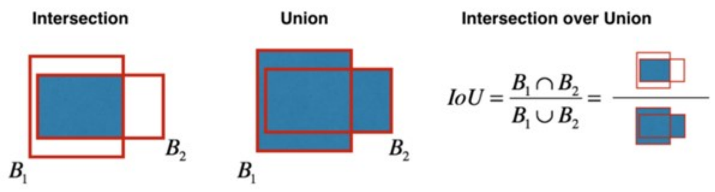
The predicted bounding box coordinates bx, by, bw, bh are obtained by applying offsets from the network output tx, ty, tw, th to the prior anchor boxes cx, cy, pw, ph as follows:

bx = (tx) + cx
by = (ty) + cy
bw = pw * exp(tw)
bh = ph * exp(th)

This transforms the anchor box priors using the predicted offsets to obtain the final bounding box predictions. The objectness score represents the probability an object is present in the box, passed through a sigmoid to get a value between 0 and 1. Class confidence values depict the probabilities of the detected object belonging to each class. Originally Softmax was used, but recently sigmoid activations were adopted to avoid the assumption that classes are mutually exclusive. For example, with an input size of 416 x 416, YOLO predicts (52x52 + 26x26 + 13x13) x 3 = 10647 bounding boxes total. YOLO transforms anchor boxes using predicted offsets and computes objectness and class probabilities using sigmoid activations, enabling fast end-to-end prediction of bounding boxes and classes.

Single Shot Multibox Detector- SSD performs simultaneous object localization and classification in a single neural network evaluation. Its architecture is based on VGG-16, leveraging the model's strong performance on high quality image classification tasks where transfer learning is applicable. The fully-connected layers of VGG-16 are discarded and replaced with additional convolutional layers (conv6 onwards). These auxiliary convolutional layers enable extraction of features at multiple scales, progressively reducing dimensionality. By building on VGG-16 and adding multiscale feature maps, SSD can predict bounding boxes and class probabilities directly from full images in a single pass. The single-shot, unified architecture allows for real-time processing without compromising too much on accuracy. SSD demonstrates how models pretrained on image classification can be repurposed for performant object detection with minor architectural modifications.(8)

The accuracy of the predicted bounding boxes can be assessed through comparison with the ground-truth bounding boxes using a metric called Intersection over Union (IoU) ((1)).



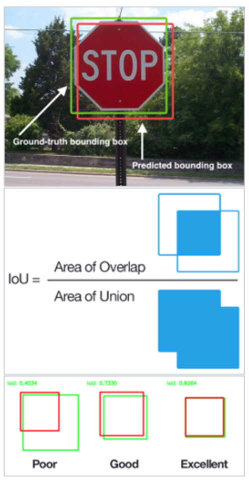

Ground-truth bounding box
Predicted bounding box
IoU = Area of Overlap / Area of Union
Poor
Good
Excellent

Figure 7: Extensive Calculation of the IoU

The region of intersection between the projected bounding box and the ground truth bounding box represents the area of interest. Meanwhile, the area of union between the two bounding boxes encapsulates the combined area covered. By calculating the IoU, which is the ratio of the intersection over union, we can quantify the accuracy of the predicted bounding boxes against the ground truth.

Non-Max Suppression works to eliminate bounding box redundancy after the YOLO model predicts multiple boxes around the same object. It follows these main steps:

1. Select the predicted box with the highest confidence score as the starting point.

2. Calculate the Intersection over Union (IoU) overlap between that box and all other predicted boxes.

3. Remove any boxes that have an IoU higher than a set threshold, indicating they substantially overlap with the selected box.

4. Repeat steps 1-3, selecting the next highest confidence box and eliminating its high-overlap neighbors.

5. Continue until no remaining boxes have a lower confidence score than the currently selected box.

This iterative process removes duplicate and overlapping boxes, leaving only the highest quality detections. Non-Max Suppression is thus a crucial post-processing step for generating clean bounding box predictions.

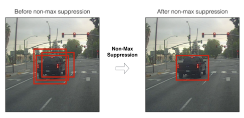


Figure 8: Non Max Suppression ((1)

4.1 Research Resources– Time Series Forecasting

Time series forecasts made using the ARIMA function are based on the autoregressive integrated moving average model. ARIMA combines autoregression for incorporating past values and moving average for including past errors. The tool fits time series data by estimating future points from historical points and error terms. It also calculates the values of several forecasting evaluation criteria.

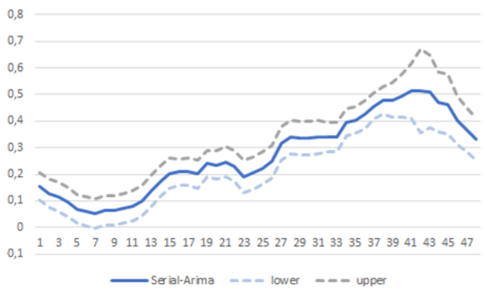


Figure 9: ARIMA Forecast

4.2 Cloud resources:

To deploy YOLO on AWS, first provision an EC2 instance with the compute resources needed to run YOLO efficiently. Be sure to configure the security groups and networking appropriately.

Next, install the YOLO dependencies and software on the EC2 instance. Transfer the trained YOLO model and any necessary data to the instance as well.

Then, execute the YOLO model on the EC2 instance, passing it the input images or video to analyze. The model will run inference and produce output bounding boxes and predictions.

Finally, retrieve the predictions from the EC2 instance either by downloading the output files or accessing them directly. The YOLO model is now deployed on AWS and ready for practical use.

5 Evaluation

The study analyzes pre-trained models for detecting six common road objects: Car, Person, Bus, Truck, Train, and Bicycle. It focuses on real-time object detection, focusing on speed and accuracy. Performance metrics include Mean Average Precision (MAP), log-average miss rate, and time taken by each model..The samples of the test set are displayed below:

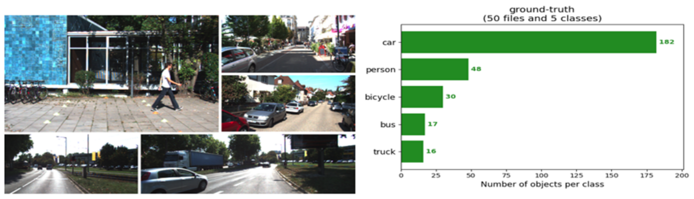


Figure 10: Number of objects per class in the dataset and Test Data Samples

5.1 Case Study 1 – Single Shot Detection Network (SSD Net)

The Single Shot Detection Network is trained on PASCAL Visual Object Classes 2014 and 2017, with 20 classes, including car, train, bicycle, person, and bus. The model has no truck class, resulting in zero Average Precision (AP). The VOC dataset contains images of various objects, and due to the low number of truck objects, they were renamed "bus" for analysis purposes. Also, the True Positives and False Positives plot has been presented in the diagram. We observe that the performance of the SSD net on detection of bicycles is poor when the image bears front or rear view of bicycles while the performance is relatively better when the image bears side view of bicycles. With the cars being most commonly seen on roads the dataset images bear a maximum number of objects belonging to the class “car”, hence, the AP is highest for the class. Also, the fraction of true positives is maximum for class “car”. Though the network identifies the car even when not completely visible or when visible in different orientations, it fails to achieve the same when the pixel size of the car is very low. As one can observe from the above image, the SSD Net fails to identify many of the cars located far from the camera. Also, as seen in the image below, the detector fails to identify the bus, which though is prominently visible. This is due to the small pixel size of the object(s). The performance of the network in detecting objects which are not well illuminated or have a color which is not in contrast with the background, the detector fails to identify the objects. It is obvious that any object with the color similar to its background can be camouflaged and not get identified in the process. There are many such examples where the object is discernible from the background when observed visually, yet not detected by SSD Net. We observe from the above histogram plot that the mean time taken by the SSD Net to detect all possible objects of classes it’s trained on, in a given image is close to 2.9s. The above plot was obtained from detection time on the first 1000 images of Kitti Dataset. Thus, we observe that, though the algorithm is relatively fast (relative to other models explored), the detection accuracy and number of instances (detections) per image is also low.

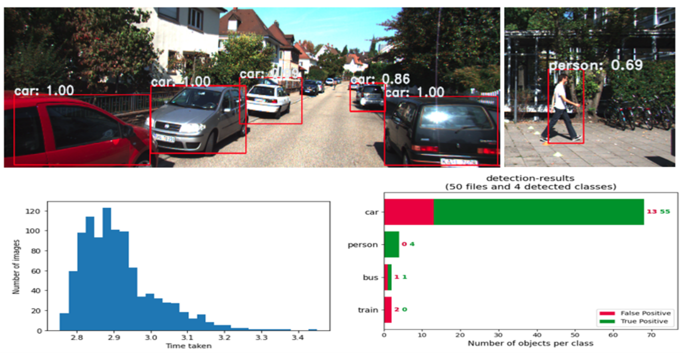


Figure 11: SSD image detection,Histogram & detection results

5.2 Case Study 2 – You Look Only Once (YOLO) v3

The You Look Only Once version 3 network is trained on the COCO dataset, including six classes: car, train, bicycle, person, truck, and bus. The average precision is plotted on the first 50 images of the Kitti dataset. Also, the True Positives and False Positives plot on first 50 images of Kitti dataset have been presented in the diagram. The YOLO v3 algorithm’s performance on bicycle detection is poor when the image has a front or rear view, but better when it has a side view, similar to SSD Net’s performance. It also performs well in detecting buses, trucks, and trains, but misclassifies large vehicles as buses. The algorithm detects people with high accuracy when the entire body is visible, but poor in cases where only the upper half is visible. The probability of detection is higher when bicycles are isolated, as it is easier to identify and locate individuals. The model, YOLOv3 is better than SSD Net in detecting the objects as well as in predicting the bounding box coordinates. The bounding box coordinates predicted by YOLOv3 are close to perfect bounding boxes. Also. the model is able to identify the objects with high confidence (¿0.7 probability), even those which are occluded partially. This can be traced back to the possible training of the mode to detect and identify class objects with occlusion's. We observe from the above histogram plot that the mean time taken by the SSD Net to detect all possible objects of classes it's trained on, in a given image is close to 4.49s. The above plot was obtained from detection time on the first 1000 images of Kitti Dataset. Thus, we observe that, though the algorithm is slow relative to SSD Net, the detection accuracy and number of instances (detections) per image is higher than that of SSD Net.

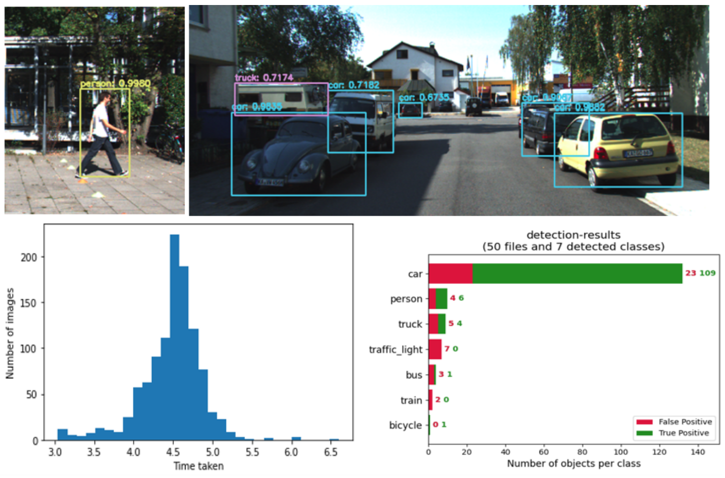


Figure 12: YOLOv3 image detection,Histogram & detection results

5.3 Case Study 3 – Mask Region-based Convolutional Neural Networks

The Mask Region-based Convolutional Neural Networks trained on Common Objects in Context (COCO) dataset. Below is the plot of Average Precision for Region-based Convolutional Neural Network on the first 50, 210 and 1000 images of Kitti Dataset. In this case, the class names were retained as it is unlike in the previous two model implementations. Also, the True Positives and False Positives plot on first 50, 210 and 1000 images of Kitti dataset have been presented below: The Mask RCNN model has poor performance in detecting bicycles, with the lowest true positives for the "bicycle" class. The algorithm's performance in detecting buses is relatively poor, but it outperforms other models in detecting trucks and trains. The best performance is found in detecting people, with bounding box coordinates close to the annotated values. The highest AP for the "person" class is also the highest among the three models, making it suitable for automated driving vehicles. However, the model's main drawback is the high detection time per image, which may affect streaming.

5.4 Case Study 4 – You Look Only Once (YOLO) v4

The You Look Only Once version 4 Network trained on Common Objects in Context (COCO) dataset. It bears 80 classes out of which all the 6 classes of our interest are included, i.e., car, train, bicycle, person, truck and bus. The COCO dataset bears images of objects in various orientations and sizes. Below is the plot of Average Precision for You Look Only Once version 4 on the custom annotated, first 50 training images of Kitti

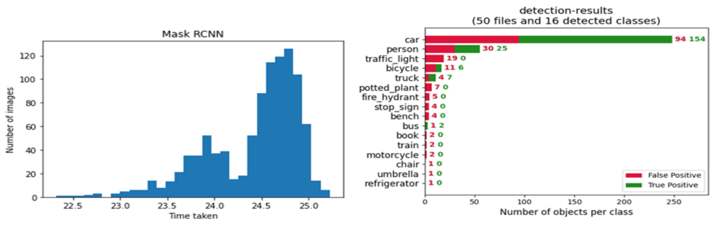


Figure 13: MASK RCNN image detection,Histogram & detection results

Dataset. Also, the True Positives and False Positives plot on first 50 images of Kitti dataset have been presented in the diagram: We observe that the performance of the YOLOv4 on detection of buses is better compared to other models in the list, though the detection of trains is very poor. As reasoned previously, even in the case of YOLOv4, the AP of class “car” is highest. The algorithm’s performance in detecting heavy duty vehicles is still poor, whereas its performance in detecting trucks is better than detection of other two classes (trains and buses) as can be TP-FP numbers for the classes in YOLOv4. The model, YOLOv4 overcomes the issues faced in YOLOv3, which is non-detection of objects far from the camera and occupying a small pixel space of the image and is able to detect even objects of classes other than the class “car” with high confidence. The samples of results obtained are displayed in the diagram, which support the same.Though this might seem like a possible candidate for application in tracking and real time object detection in traffic, the most crucial drawback of this model is the high time taken for detection per image, which translates to high detection time per frame in streaming. We observe that the performance of the YOLO v4 net on detection of bicycles is poor when the image bears front or rear view of bicycles while the performance is relatively better when the image bears side view of bicycles. With the cars being most commonly seen on roads the dataset images bear a maximum number of objects belonging to the class “car”, hence, the AP is highest for the class. Also, the fraction of true positives is maximum for class “car”.

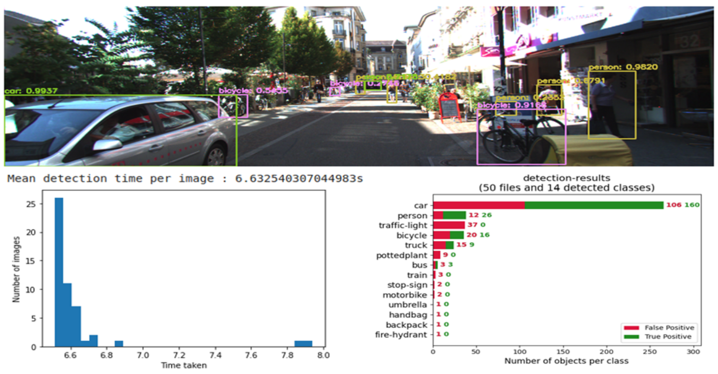


Figure 14: YOLOv4 image detection,Histogram & detection results

5.5 Case Study 5 – Retina Net (ResNet50)

In this section, we describe the results obtained on testing the pre-trained Retina Net model, trained on the Open Images Dataset. The model is trained on 500 different classes of Open Images Dataset out of which all the 6 classes of our interest are included, i.e., car, train, bicycle, person, truck and bus. Below is the plot of Average Precision for Retina Net on the custom annotated, first 50 training images of Kitti Dataset. The class names were retained as it is unlike in the previous two model implementations. Also, the True Positives and False Positives plot on first 50, 210 and 1000 images of Kitti dataset have been presented in the diagram. The Retina Net outperforms SSD Net and YOLO v3 in detecting bicycles, possibly due to its use of stationary and non-stationary classes during training. This helps the model identify features of a complete bicycle and distinguish between stationary and seated bicycles, improving detection accuracy. The algorithm's performance in detecting buses and trains is relatively poor, while its performance in detecting trucks is better. ResNet50 is comparable to YOLOv4, but its performance in detecting low illumination and occlusion is poorer. Additionally, the model's detection of bicycles piled up is low, unlike YOLOv4, which performs better than Retina Net. Though this might seem like a possible candidate for application in tracking and real time object detection in traffic, the most crucial drawback of this model is the high time taken for detection per image, which translates to high detection time per frame in streaming.

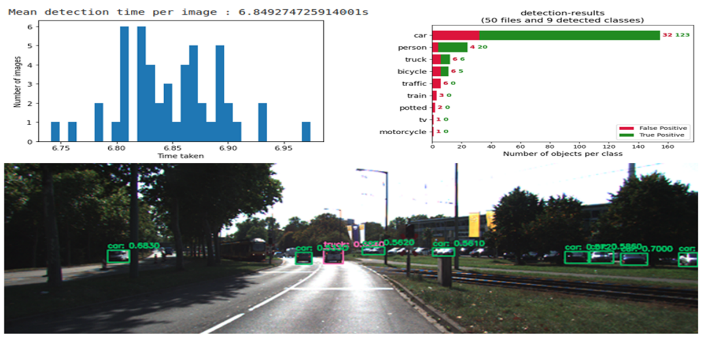


Figure 15: RetinaNet image detection,Histogram & detection results

5.6 Case Study 6 – Testing on Real Time Data

Before testing the algorithm on BSCL data, there were two important things to be taken care of - retraining and increasing speed of processing. Since the pre-trained model cannot detect custom Indian vehicles, it has to be retrained for detecting vehicles like auto-rickshaws, motorbikes, etc. Secondly we have to speed up the process immensely by taking GPU into account, making the algorithm fit for real time scenarios and traversing through large volume of data for analysis. For the first problem, data was annotated from frames provided by BSCL and fed into the system for retraining using Darknet. To speed up the process, custom PyTorch algorithm was written which gave an outstanding 18 fps detection (which is approximately 34 times faster than only CPU usage). There was a considerable improvement in object detection of Indian vehicles after retraining:

The mAP found using the pre-trained model was 24.31%. The important class to observe is rickshaw. The pre-trained best model selected from the analysis didn't performa

	Class	AP(%)
0	car	58
1	bus	41
2	truck	21
3	person	15
4	bike	11
5	rickshaw	0

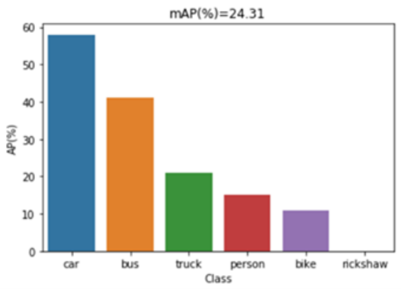


Figure 16: Performance using pre-trained model on BSCL Dataset

as expected. Hence we present the re-trained module of YOLOv3 on Indian streets. The mAP = 31.04% which is much improved from the previous case.

	Class	AP(%)
0	car	58
1	bus	51
2	truck	30
3	person	15
4	bike	16
5	rickshaw	21

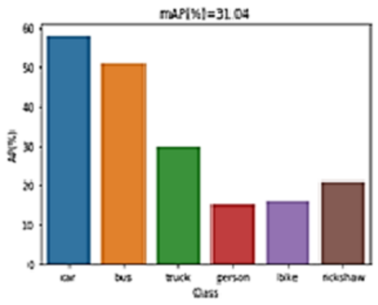
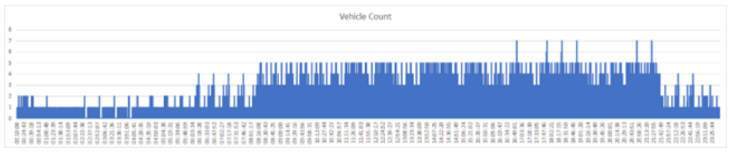


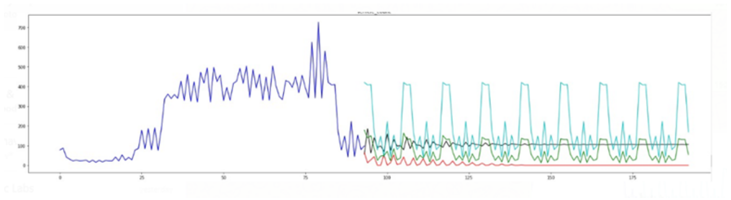
Figure 17: Performance of Retrained model (Indian – YOLOv3)

5.7 Case Study 7 – Time Series for Green Time signal

The data that is used here is of 15th September 2019. This is a per second dataset which is the representation of the number of vehicles coming to one roadside at Rupali Square Bhubaneswar. Although this is a sample dataset which is used for building the time series forecasting model, the real dataset is going to be used in the real time from the Object Detection YOLOv3 with SORT algorithm. Sample Plot Sample Result Plot

	Time Elaspsed	Vehicle Count
0	2019-09-15 09:40:12	0
1	2019-09-15 09:40:13	0
2	2019-09-15 09:40:14	0
3	2019-09-15 09:40:15	0
4	2019-09-15 09:40:16	0
5	2019-09-15 09:40:17	0
6	2019-09-15 09:40:18	1
7	2019-09-15 09:40:19	1
8	2019-09-15 09:40:20	1
9	2019-09-15 09:40:21	0





The result is from different models and the complexity of the model that is being done is reviewed in the following sections.

6 Discussion

6.1 Object Detection Model Evaluation

This research evaluated six object detection models on the KITTI dataset for traffic monitoring: SSD, YOLOv3, Mask R-CNN, YOLOv4, RetinaNet, and a custom YOLOv3 model.
YOLOv4 achieved the best balance of accuracy and speed. With 42.44% mean Average Precision, it outperformed other models except Mask R-CNN. It detected cars effectively with 79.3% AP. However, YOLOv4's bicycle detection was lacking. This could be improved by training on more bicycle data.
Mask R-CNN had the highest accuracy at 46.7% but was too slow for real-time use, taking over 4 seconds per image. This highlights the tradeoff between accuracy and speed.
RetinaNet was less accurate and slower than YOLOv4. Its main advantage was stable processing times.
The simple SSD model was very fast but lacked accuracy. The custom YOLOv3 model performed much better than the pre-trained version, showing the importance of customization.

6.2 Cloud Computing for Scalable Real-Time Processing

To enable real-time processing of video streams from many edge cameras spread across the city, the system leveraged cloud computing on AWS EC2 instances with GPU acceleration. This provided the scalable processing power needed for low-latency analysis, which would be difficult with dedicated hardware alone. The hybrid edge-cloud architecture allowed initial computer vision at the edge before aggregated data was sent to the cloud for coordination.

6.3 Limitations

Some limitations exist. The models were only evaluated on 50 KITTI images. Larger and more diverse datasets could provide more confidence.
Testing on video rather than images could give a better sense of real-world performance.

Deploying on embedded systems would reveal true speed and accuracy metrics.

7 Conclusion and Future Work

YOLO is a fully convolutional network (FCN) with 75 convolutional layers, skip connections, and up sampling layers. It uses a convolutional layer with stride 2 to down sample feature maps, preventing loss of low-level features. YOLO is invariant to input image size, but for batch processing, fixed height and width images are needed. YOLOv3 with a GPU (Nvidia Tesla K80) is preferred for computation.

7.1 Salient Features

1. Works fast, extremely suitable to real time application.
2. Algorithm can identify custom Indian vehicles like auto rickshaws, motorbikes etc.
3. Deployment on edge devices.
4. Comes with edge/video analytics.
5. Optimization of traffic lights via Reinforcement Learning.

7.2 Future Work

Some areas for future work include:

1. Detecting emergency vehicles and incorporating priority logic into the system.
2. Exploring ways to further optimize system performance and accuracy.
3. Connecting multiple traffic signals to enable city-wide optimization.
4. Testing the system on video data for more real-world validation.
5. Deploying the system on embedded hardware at scale.

7.3 Concluding Statement

Research shows reinforcement learning algorithms improve traffic flow at intersections, enabling policy mapping without rule-based rules or events. This system demonstrates the potential of an integrated edge-cloud approach for real-time traffic monitoring and adaptive signal control to reduce congestion.

References

[1] Bochinski, E., Senst, T. and Sikora, T. (2018). Extending iou based multi-object tracking by visual information, *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6.

[2] Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection.

[3] Chaulya, S. and Prasad, G. (2016). Chapter 2 - mine transport surveillance and production management system, *in* S. Chaulya and G. Prasad (eds), *Sensing and*

Monitoring Technologies for Mines and Hazardous Areas, Elsevier, pp. 87–160.
URL: *https://www.sciencedirect.com/science/article/pii/B9780128031940000027*

[4] Chen, Y., Li, W., Sakaridis, C., Dai, D. and Gool, L. V. (2018). Domain adaptive faster r-cnn for object detection in the wild.

[5] do Nascimento, N. M. and de Lucena, C. J. P. (2017). Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things, *Information Sciences* **378**: 161–176.
URL: *https://www.sciencedirect.com/science/article/pii/S0020025516313664*

[6] Hernández, J. Z., Ossowski, S. and Garcıa-Serrano, A. (2002). Multiagent architectures for intelligent traffic management systems, *Transportation Research Part C: Emerging Technologies* **10**(5): 473–506.
URL: *https://www.sciencedirect.com/science/article/pii/S0968090X02000323*

[7] Liu, C. and Ke, L. (2023). Cloud assisted internet of things intelligent transportation system and the traffic control system in the smart city, *Journal of Control and Decision* **10**(2): 174–187.
URL: *https://doi.org/10.1080/23307706.2021.2024460*

[8] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016). Ssd: Single shot multibox detector, *in* B. Leibe, J. Matas, N. Sebe and M. Welling (eds), *Computer Vision – ECCV 2016*, Springer International Publishing, Cham, pp. 21–37.

[9] Peng, Q., Luo, W., Hong, G., Feng, M., Xia, Y., Yu, L., Hao, X., Wang, X. and Li, M. (2016). Pedestrian detection for transformer substation based on gaussian mixture model and yolo, *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)* **02**: 562–565.
URL: *https://api.semanticscholar.org/CorpusID:17550989*

[10] Rath, M. (2018). Smart traffic management system for traffic control using automated mechanical and electronic devices, *IOP Conference Series: Materials Science and Engineering* **377**(1): 012201.
URL: *https://dx.doi.org/10.1088/1757-899X/377/1/012201*

[11] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement.

[12] Ren, S., He, K., Girshick, R. and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks, *in* C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems*, Vol. 28, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper*$_f$*iles/paper/*2015/*file*/14*bfa*6*bb*14875*e*45*bba*028*a*21*ed*3
Paper.pdf

[13] Sommer, L., Schmidt, N., Schumann, A. and Beyerer, J. (2018). Search area reduction fast-rcnn for fast vehicle detection in large aerial imagery, pp. 3054–3058.

[14] Sun, X., Wu, P. and Hoi, S. C. (2018). Face detection using deep learning: An improved faster rcnn approach, *Neurocomputing* **299**: 42–50.
URL: *https://www.sciencedirect.com/science/article/pii/S0925231218303229*

[15] Ullah, A., Xie, H., Farooq, M. and Sun, Z. (2018). Pedestrian detection in infrared images using fast rcnn, pp. 1–6.

[16] Wan, S., Ding, S. and Chen, C. (2022). Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles, *Pattern Recognition* **121**: 108146.
URL: *https://www.sciencedirect.com/science/article/pii/S0031320321003332*

[17] Wang, X., Shrivastava, A. and Gupta, A. (2017). A-fast-rcnn: Hard positive generation via adversary for object detection.

[18] Zhang, S., Yao, Y., Hu, J., Zhao, Y., Li, S. and Hu, J. (2019). Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks, *Sensors* **19**(10).
URL: *https://www.mdpi.com/1424-8220/19/10/2229*

[19] Zhou, S., Wei, C., Song, C., Pan, X., Chang, W. and Yang, L. (2023). Short-term traffic flow prediction of the smart city using 5g internet of vehicles based on edge computing, *IEEE Transactions on Intelligent Transportation Systems* **24**(2): 2229–2238.

Reference Title or Description