DEEP LEARNING FOR EXPLAINABLE TRAFFIC ANOMALY DETECTION IN DHAKA

Submitted By

Samia Haque Tisha
ID: 201-35-2973
Department of Software Engineering
Daffodil International University

Supervised By

Md. Shohel Arman
Assistant Professor
Department of Software Engineering
Daffodil International University

This Thesis report has been submitted in fulfilment of the requirements for the Degree of Bachelor of Science in Software Engineering.

Spring 2024

APPROVAL

This thesis titled on **"Deep Learning for Explainable Traffic Anomaly Detection in Dhaka"**, submitted by **Student Name (ID: 201-35-2973)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. Imran Mahmud
Associate Professor & Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Md. Maruf Hasan
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Md. Shohel Arman
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Dr. Md. Sazzadur Rahman
Professor
Institute of Information Technology
Jahangirnagar University

DECLARATION

I hereby declare that, this thesis report is done by me under the supervision of Md. Shohel Arman, Assistant Professor, Department of Software Engineering, Daffodil International University. I, therefore, state that neither this thesis nor any part has been submitted else here for the award of Bachelor's degree or any graduation.

Supervised By

Md. Shohel Arman
Assistant Professor
Department of Software Engineering
Daffodil International University

Submitted by

Samia Haque Tisha
ID: 201-35-2973
Department of Software Engineering
Daffodil International University

ACKNOWLEDGEMENT

First of all, I want to thank Almighty God for His divine favor, enabling me to complete my undergraduate thesis. It is driven by the interest in how to leverage the potentials of deep learning and explainable AI to address pressing real-world problems, particularly in the domain of urban development and traffic management.

I would express my deepest sense of thanks to my thesis supervisor, Md. Shohel Arman, Assistant Professor of the Department of Software Engineering, for his guidance and all-out support given to me in the entire research work. His intellectual advice and insights have largely shaped this work, and his commitment without wavering has been the inspiration for me to explore the limits of my knowledge and skills.

I would like to express my thanks to Dr. Imran Mahmud, Head of the Department of Software Engineering, Faculty of Science and Information Technology, and my other professors, faculties, and personnel for their kind cooperation and support in the successful completion of my work.

Great thanks to my parents and friends for continuous support, patience, and understanding during these years. Their support has been my stronghold, letting me have the opportunity to weather the turmoil that I went through.

Finally, I would like to acknowledge my batchmates and fellow members of DIU for their kind cooperation and consolation, which helped me reach this goal, as well as the organizations providing data and resources necessary for the research work. Without them, this work was not possible.

ABSTRACT

Dhaka suffers from severe traffic anomalies, whereby congestion affects not only economic productivity but also quality of life, making it the slowest city in the world because of its severe traffic problems. Despite the widespread impact, existing deep learning studies are limited and leave valuable insights untapped, requiring internet connectivity and offering limited real-time analysis of images or live video feeds. I addressed these challenges in a multi-stage Traffic Congestion Region Detection framework that identifies a congested region with object counting, fusing object detection with color-coded bounding boxes and connected components. This is a junction of state-of-the-art, advanced deep learning models for object detection with Explainable AI techniques that provide explanations of model decisions through detailed visuals, hence improving transparency. Since this research is related to working with one of the world's most congested traffic zones in Dhaka, my methodology consisted of extensive data preparation by doing exploratory data analysis and data augmentation for model generalizability with traffic images. The TCRD framework achieved high detection accuracy with the YOLOv10 model, among both YOLOv9 and YOLOv10, attaining a 94% mAP, 94% precision, and 91% recall score. Employing Eigen-CAM confirmed the accuracy, transparency and potential for improvement in model decisions. In particular, this new multi-stage framework can really identify the congestion regions of different categories in an image and give insights into other traffic anomalies, hence hugely stepping toward the improvement of traffic management solutions in urban environments like Dhaka. As a result, my solution can be integrated into enhanced traffic management and urban planning strategies to realize more efficient and effective solutions for traffic anomalies.

Keywords — Deep Learning, Explainable AI, Traffic Anomaly, Dhaka, Traffic Congestion, YOLOv9, YOLOv10, Eigen-CAM, Traffic Congestion Region Detection, TCRD, Object Detection, Multi-Stage Framework, Traffic Analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In the case of Dhaka, traffic anomalies in most cases are huge congestions that derogate from the normal trend, affecting daily commutes and thus the quality of life in general. What used to be an occasional hassle on main city streets escalated into a general problem affecting even residential areas. Although roughly Tk 135,000 crore has been spent on infrastructure megaprojects like flyovers and metro rail, there is no relief from traffic congestion as Dhaka was identified by the US National Bureau of Economic Research as the slowest city globally (Mustafa, 2023). It also comes with an economic cost, which is approximately 5 million work hours daily and causes a loss of USD 11.4 billion annually (Haider, 2018). Such chaotic scenes might give way to serious physical and mental health issues, including raised levels of stress and aggression (Haider, 2018). Evidently, one of the major anomalies in the urban setup of Dhaka is traffic congestion, for which some innovative methods of management are long overdue. My system detects vehicles and pedestrians in images of traffic using advanced state-of-the-art deep learning models to identify congestion through a fully colored region search, based on the count of the objects. Integrating techniques from Explainable AI solves the "black box" problem by providing a visual explanation of the anomalies that were detected for transparency in traffic management. This would not only offer very valuable insight into the area of traffic management but also shed some very important light on when there is a pressing need for effective solutions.

1.2 BACKGROUND

The worst traffic situation has been going on in Bangladesh, much more in Dhaka, for the past years, and has reached a chaotic state of traffic. For example, the average speed of the traffic in Dhaka was 21 km/h back in 2007, which drastically dropped to 4.8 km/h in 2022 (Mustafa, 2023). This steep fall in traffic speed portrays the severe congestion that visits the city every day. Moreover, huge human costs are paid due to traffic incidents all over Bangladesh, killing at least 64 people and injuring 150 others daily (Haider, 2018). In the middle of the roads, intercity buses and trucks normally park and offload goods onto the streets, which worsens the chaotic traffic (Mustafa, 2023). Several existing solutions have been proposed and implemented in respect of

these challenges. These include real-time traffic updates, based on GPS, satellite, and user data, managing and mitigating traffic congestion to some extent. Deep learning methods have been applied to classify image data in an attempt to understand the patterns of the traffic. Deep learning includes a number of multiple layers of neural networks modelling complex patterns in data (Borgalli and Surve, 2023), including explainable AI focusing directly on the understandability of AI decisions for humans (Coussement et al., 2024). Diverse data prevents overfitting and maintains great accuracy for all possible outcomes achieved by deep learning. However, most of the methods fall short because they do not directly analyze images to give insights into congested regions, which is very important in the effective management and planning of traffic.

1.3 PROBLEM STATEMENT

To many of us, congestion in Dhaka has become part of our daily harassment. This is a big disturbance to normal life and causes abnormal conditions for many more. Studies suggest that the average commuter of Dhaka city spends 46 minutes every two hours stuck in traffic, amounting to approximately 276 hours of annual wastage per commuter (Mustafa, 2023). The inequality in the use of road space is very sharp: 6% of private car commuters occupy 76% of the road space, driven by infrastructure developments catering to private car owners (Mustafa, 2023). This prioritization packs more cars onto the streets and, hence, adds to urban congestion, undercutting the use of public transport. Unhandled problems with vehicle management, infrastructure, and the traffic system lead to daily traffic anomalies; congestion is the most central anomaly, interlocked with all others like accidents, traffic violations, illegal parking, etc. as I portrayed in Figure 1.1.

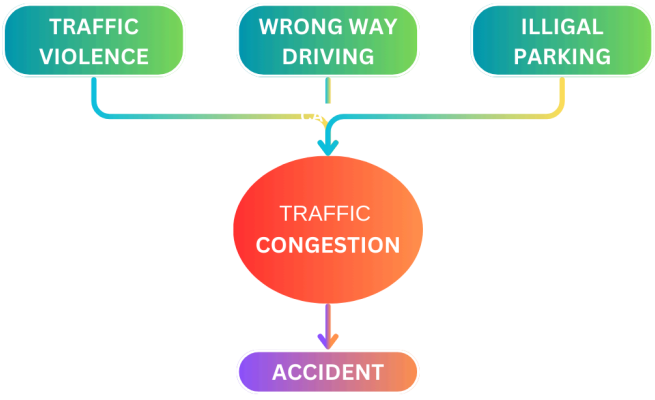

Figure 1.1: Traffic Congestion as Central Traffic Anomaly.

Despite the fact that traffic congestion is very prevalent and has huge impacts, existing deep learning researches have their own limitations. The current studies in the literature specifically

categorize images only into light and heavy congestion without getting into detailed analysis of congestion. It just ignores the very fact that there are certain congested regions in the images that need to be analyzed to provide actionable insights. Furthermore, while tools such as Google Maps do detect congestion on route maps, they require an internet connection and support limited analysis of each image or live video feed in real time—leaving many insights buried.
Dhaka Traffic Management Needs Advanced Techniques to Analyze Images with Congested Region.

1.4 RESEARCH GAPS

In existing systems, there is a lack of undertaking comprehensive analyses of congested regions using real-time data or imagery, causing a wide gap in gaining meaningful insights. The existing methods do not give full insight into the points of congestion and what the root cause is, making traffic management strategies ill-informed. Therefore, gives an urgent call for the incorporation of Explainable AI in order to shed more light on why congestion occurs and also enhance transparency. The inadequate application of advanced deep learning models for detail congestion detection adds to the problem and points out a very critical area of research and development. Addressing these gaps will significantly improve the efficacy of traffic management solutions in urban environments like Dhaka.

1.5 OBJECTIVES

The objective of my study is-

1. **Multi-Stage Traffic Anomaly Analysis Framework:** I aim to develop a comprehensive analysis framework that combines object detection, coloring bounding boxes, finding connected components for fully colored regions, and detecting congested regions by object counting. This framework shows clearly that many traffic anomalies prevail and provides a clear insight into the locations of congestion, if any, in the images of Dhaka city.
2. **Integration of Traffic-Specific Explainable AI:** I will strive to incorporate the traffic-specific explainable AI methods, specifically the Eigen-CAM, to generate detailed visual explanations of model decisions towards the required transparency needed for the different applications related to traffic management.

1.6 MOTIVATION OF THE STUDY

My interest in this study is highly motivated by the daily life and economic activities disruptions resulting from traffic congestion. There's a clear gap in the absence of systems that can analyze congested regions either from an image or real-time data, which prevents effective management of traffic. Some of the insights to be drawn from the analysis of the congestion can be vital in avoiding accidents and other traffic anomalies. I think that if advanced traffic management techniques are adopted, by as high as 40%, congestion may be reduced (Haider, 2018); hence, tremendous improvement in quality of life and economic productivity could be assured in urban settings like Dhaka.

1.7 SUMMARY

In this chapter, I have introduced the severe problems of traffic congestion in Dhaka, emphasizing how it has affected daily life routines and economic activities. Infrastructural projects involving huge investments so far have not been able to alleviate congestion, which is still a critical problem and causes massive economic losses, besides affecting public health adversely. I have also explained the inadequacy of the present management system with regard to analyzing congested regions from real-time data or images. These systems are further non-transparent and hence incapable of providing fine-grained insights to manage and mitigate congestion. I identified key gaps in research that incorporate the need for advanced deep learning models and Explainable AI techniques to explain comprehensive congestion analysis, leading to actionable insights. Finally, the motivation and objectives of my study are motivated by possible gains that enhanced traffic management techniques could bring about in terms of reducing congestion and improving quality of life within urban settings such as Dhaka.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Unprecedented urban population growth with a corresponding explosion of vehicles in cities from around the globe, and countries like the capital city of Bangladesh, Dhaka, have presented challenges for a safe and sustainable traffic flow. It is characterized by traffic jams and incidents of road mishaps.

Traffic congestion is a deep-rooted problem that affects many aspects of urban life, from our patterns of commute to economic efficiency, environmental sustainability, and social well-being. Dhaka is the slowest city in the world, with traffic speed dropping from 21 km/h in 2007 to 4.8 km/h in 2022, as reported by the US-based National Bureau of Economic Research (Mustafa K., 2023). Despite decades of spending lavishly on highways, the congestion results from the nearly exclusive dedication of space to private cars at the expense of public transportation (Mustafa K., 2023). Bangladesh's capital Dhaka ranked 5th worst in traffic worldwide. The ranking of the most congested cities in the world is given by the Tribune Desk report, with the rapid growth in urban population and corresponding vehicle rise (Tribune Desk, 2023).

A vibrant literature review on traffic jams in Dhaka alongside other cities offers a more comparative viewpoint to address the problem. It will allow a better understanding of existing knowledge regarding this problem and, by identifying its shortcomings, also help in suggesting and developing solutions that are innovative and directly applicable to the problems of Dhaka.

2.2 PREVIOUS LITERATURE

Rahman et al. developed a deep hybrid model (ConvLSTM) with attention mechanisms for traffic flow prediction, which outperforms other models, but this model is not interpretable (Rahman et al., 2023). Rahman M.M. and Nower N. (2024) used Dynamic Time Warping(DTW) and clustering for spatiotemporal traffic pattern analysis in Dhaka, but these methods did not use real-time data or advanced machine learning models. Sundas et al. used deep learning-based UAVs for traffic congestion control using One-stage detectors YOLO, YOLOv3 and YOLOv4 and two-stage detectors Faster R-CNN and Cascade R-CNN with 93% accuracy in 2023 but lacked interpretability of the model (Iftikhar et al., 2023). Bindu et al. applied CNN for traffic congestion detection in surveillance video, achieving an accuracy of 93%, but did not operate with real-time data processing (G. Bindu Madhavi et al., 2023). Tan et al. implemented vehicle

tracking with YOLOv5 and Deep SORT, achieving mAP@0.5/0.93 but did not address low-light conditions (Tan & Kieu, 2023). Zunair et al. applied the RSUD20K dataset for autonomous driving intelligence, achieving 77.9% mAP with YOLOv6-M6 but not targeting traffic congestion detection (Zunair et al., 2024). Adnan et al. used colour traffic video classification with DCNN, achieving 98.2% accuracy but lacking generalization and explainable AI (Adnan et al., 2022). Hossain and Nower (2022) utilized KNN in traffic intensity forecasting using unpaid Google Maps information, nevertheless, the accuracy of the methods and automated models are not ensured (Hossain and Nower, 2022). Ahad et al. addressed congestion analysis in Dhaka using DFS in real-time traffic data, but it was not a deep learning model and missed the feature of explainable AI (Ahad et al., 2021). Kurniawan et al. used CNN for traffic congestion detection from CCTV feeds with 89.50% accuracy, but it lacked detailed and coloured image analysis and explainability (Kurniawan et al., 2018). Bawaneh et al. defined a detector for traffic-congestion detection using the voting ensembling method in smart cities but lacked further model testing and refinements (Bawaneh & Simon, 2023). Nidhal et al. proposed real-time traffic congestion detection using image processing with 99-100% accuracy, but it was likely to overfit (Nidhal et al., 2014). Maksudur Rahman et al. examined traffic patterns through clustering and regression using GPS data, but real-time data integration and advanced models were lacking in their work (Rahman et al., 2018). Chakraborty et al. increased accuracy 91.4%, 90.2%, and 85.7% accuracy respectively in detecting traffic congestion, and trained YOLO, DCNN, and SVM, but did not assess overfitting (Chakraborty et al., 2018). Sardar Khan et al. proposed using CNNs such as ResNet50 for traffic anomaly detection with up to 95.6% accuracy but lacked diverse data and explainable AI (Khan et al., 2022). While they use different methods to detect traffic congestion and anomalies, they have some common limitations, including real-time data usage, explainable AI, and applicability to the traffic characteristics of Dhaka as a developing country.

2.3 SUMMARY

The literature review conducted provided knowledge of various methodologies and models used in traffic detection and anomalies, focusing on Dhaka and the global context. Our review found a broad spectrum of methods, ranging from deep learning approaches to data collection strategies, but it also sheds light on some consistent gaps. Among these were the use of real-time data, explainable AI, and adaptation to Dhaka's traffic conditions. These insights offer a useful context for our research and highlight the limitations we aim to overcome, ensuring our solutions are empirical and contextually appropriate for Dhaka.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 INTRODUCTION

In this section, I present the methodology applied in deep learning-based traffic congestion detection. In this section, a dataset is prepared with 9900 images drawn from various sources, along with exploratory data analysis and its preprocessing and augmentation techniques. I exploited the newest pre-trained SOTA models, YOLOv9 [21] and YOLOv10 [22], as the object detectors and developed a bespoke framework which shall identify and classify the congestion level within the detected region. It involves training, evaluation, inference, and further visual explanation with the help of CAM techniques in the process. I intend to effectively detect traffic congestion with insights into how the model is making its decisions.

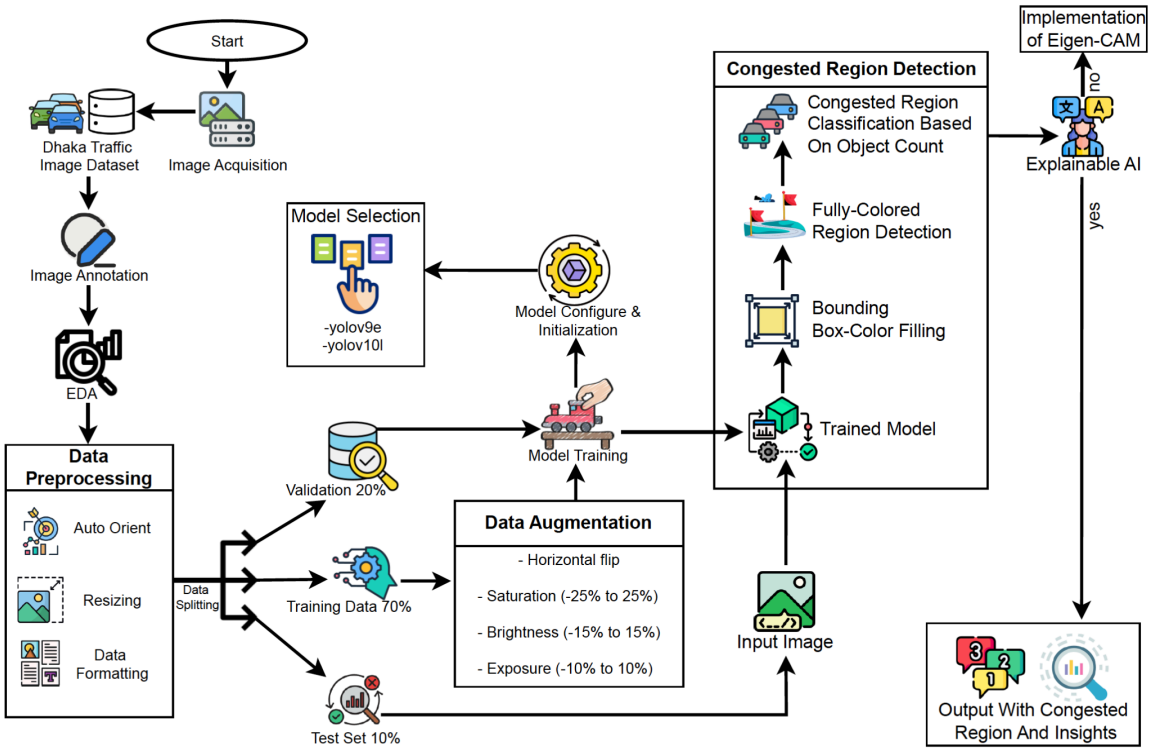

Figure 3.1: Model Architecture.

Full Text: I acquire images of Dhaka traffic with annotation, do data pre-processing, and split into train, test, and validation sets. One of these models, either YOLOv9e [21] or YOLOv10l [22], will be chosen in the next step, followed by configuration and training with augmented data. At last, it detects the bounding boxes for the input image, fills them with color, and identifies congested regions on the basis of object count. It also visualizes these areas with insights in the output; simultaneously, Eigen-CAM [23] is used for explainable AI toward transparency and accountability.

3.2 DATA PREPARATION

The quality, diversity, and readiness of the data obscure truth be told, how definitely a profound model works and even the first system of a profound model. I had gone through a process of several key stages to appropriately prepare my dataset so that it can be able to detect good traffic anomalies in Dhaka in this study. Every step is important to be able to make a model which learns well from training data but also does well with new unseen data.

3.2.1 DATA COLLECTION

In this research, I have combined the data from different sources with the data generated from my effort to make the data as rich as possible and put together a detailed data set for the elaboration of the diverse nature of traffic in Dhaka. This dataset merges 7,447 images from the Bangladeshi Traffic Flow Dataset composed of areas like Arambag and Shaplachattar (Islam, 2024), with 1700 images from the Dhaka AI dataset (Shihavuddin and Rashid, 2020). On top of that, I added 753 pictures of Jatrabari, Mirpur, Dhaka New Market, Shekertek, Dhanmondi, Mohammadpur and Shymoli myself. Pooling them all together, a dataset of close to 9,900 images was present, with a variety of complex and simple traffic scenes including day and night environments to provide a comprehensive analysis solution. Every image in the dataset is accurately annotated with 9 object categories that are “Bike”, “Bus”, “Car”, “Cng”, “Cycle”, “Mini-Truck”, “People”, “Rickshaw” and “Truck”-all of which are the prerequisites for fine-grained object detection which is essential for traffic abnormality analysis.

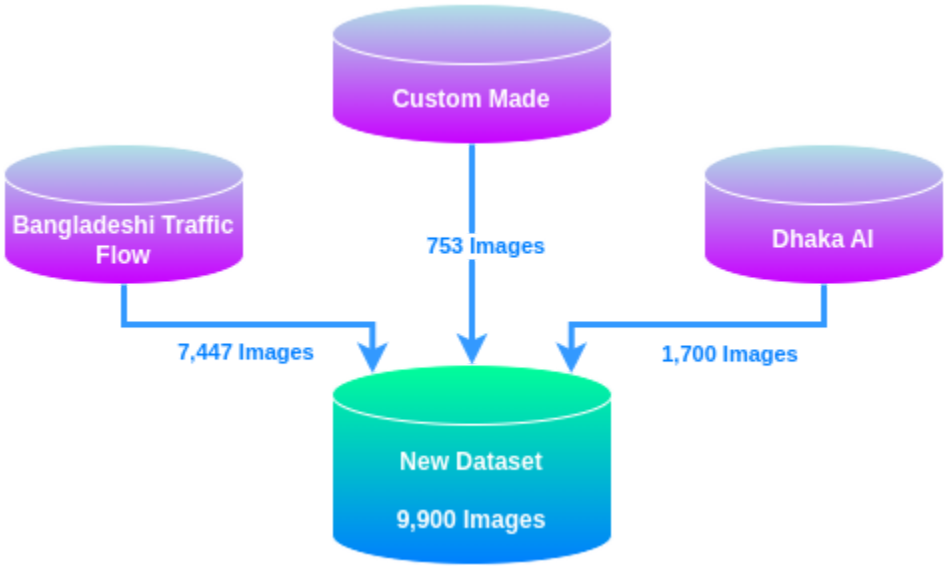

Figure 3.2: Dataset Distribution.

3.2.2 DATA PREPROCESSING

The images are preprocessed before they are fed into the model for training to make it learn the feature efficiently. To guarantee uniformity, get these images ready for the neural network; here are the preprocessing techniques applied to the dataset:

- **Resizing:** All the images were resized to 640x640 pixels to standardize their size throughout the dataset [21]. Therefore, this stage is very important to make the dimensions of the inputs consistent. That can not only make training easier but also result in much better results with fewer errors.
- **Auto Orient:** This is a step for proper orientation, ridding any EXIF rotations [24]. It makes sure that whatever device is used and whatever setting it is on, images will be correctly oriented [24]. Standardization of image orientation is desirable in avoiding potential mistakes in the training phase and hence makes the model more robust [24].
- **Normalization:** This was to ensure that the pixel values of images of the traffic are within a range from 0 to 1 [23]. This was ensured by first converting them into float32 before division for every pixel value by 255 [23]. Normalization is very important to explainable AI, in terms of maintaining scales' consistency of pixel values. This assists in generating more interpretable explanations from a model.
- **Formatting:** This depends on the model one is using and how the dataset should be formatted. Since I use the YOLOv9 [21] and YOLOv10 [22] models, I formatted the dataset for use in the YOLOv9 data structure. Having images and their annotations organized in a certain way allows the YOLO models to train effectively and allow them to do inference on these images.

These images are then saved in their pre-processed form. Hence, the data is prepared for quick loading and consumption during training, which turns into quicker convergences and good generalization.

Figure 3.3: Comparison between Original Images and Preprocessed Images.

3.2.3 DATA SPLITTING

The dataset is splitted into train (70% of the data), validation (20% of the data) and test (10% of the data) set [25]. This split allows for a large dataset that can be used for training and a sufficient amount of data left to validate and test our model, important for determining the extent to which our model generalizes and performs in real-world conditions.

3.2.4 EXPLORATORY DATA ANALYSIS

Before training models, I did a lot of exploratory work to become familiar with the characteristics of the dataset and data stability. Here I will show how I did some of those preliminary analyses in each split:

1. **Sample Image Inspection**

   I got one of the images from the training set to check its type, shape, and get an idea about its dimension. The type of the image was <class 'PIL.JpegImagePlugin.JpegImageFile'>, while the shape was (640, 640, 3), thus representing cells on both height and width of 640 pixels each with 3 color channels representing the RGB. I further did the visualization of the picture and confirmed that it is well represented.

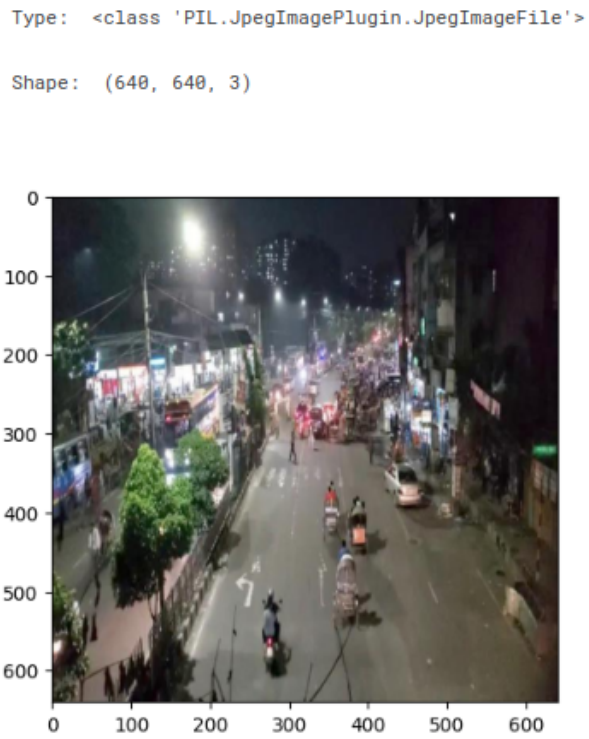

Figure 3.4: Sample Image Inspection from Train Set.

2. **Image Properties**

I viewed the properties for the sample image. These included the following:

Height: Number of pixels on the vertical axis.

Width: Number of pixels on the horizontal axis.

Channels: Number of color channels; for example, 3 for RGB.

Data Type: Type of data used in representing each pixel value; for example, uint8 reflects 8-bit unsigned integers.

```
{'width': 640, 'height': 640, 'channels': 3, 'dtype': dtype('uint8')}
```

Figure 3.5: Output of Sample Image Properties from Train Set.

3. **Set-wise Visual Inspection**

I have checked a random sample of 5 images from each set: train, validation, test, along with 5 annotated images from the training set. This step makes sure that the images and their annotations load and view correctly.

Figure 3.6: Output of Set-wise Visual Inspection.

4. **Analysis of the Distribution of Classes**

I looked through the class distribution of the dataset. I have created a table with the number of examples in every class for the train, validation, and test sets by volumes. I also used the Seaborn Library [28] to visualize the bar plot for class distribution in each set. This analysis returned imbalances with respect to the classes "Cycle", "Mini-Truck", and "Truck" in every set. These imbalances, I noted, have to be taken care of to meet the prerequisites of a balanced model training.

Figure 3.7: Analysis of the Distribution of Classes and Class Imbalances.

5. **Consistency of Image Sizes:**

I checked the sizes of images in all sets, train, validation, test, for checking uniformity. It was confirmed that all images maintained a consistent size of 640x640 pixels.

```
Image sizes in train set:
(640, 640)


Image sizes in valid set:
(640, 640)


Image sizes in test set:
(640, 640)


CPU times: user 2.25 s, sys: 303 ms, total: 2.55 s
Wall time: 2.56 s
```

Figure 3.8: Consistency of Image Sizes.

In this detailed EDA, I went through the structure and characteristics of the dataset that helped shed light on some key insights and possible areas where data balancing would occur. This provided a solid base for the running and training of the models.

3.2.5 DATA AUGMENTATION

To address the data imbalances in my dataset, I applied data augmentation only on the training set, since it is conventional, and avoided a validation set and test set to prevent contamination.

- **Horizontal flip:** Mirrored images from the actual data are created so that the model learns to be orientation-agnostic in class identification.
- **Saturation changing ranging from -25% to +25%:** Decreasing or increasing saturation in the given range changes color intensities, which makes the model resilient against different levels of vibrant colors.
- **Brightness shift from -15% to +15%:** Varied brightness makes the images lighter or darker, improving model strength for varied lighting conditions.
- **Exposure -10% to +10%:** A small change in exposure amount will slightly change the exposure level of the image and may let the model generalize across images taken under different settings of exposure.

One main reason for this augmentation is that it makes a model generalize and work well on data not seen by the model [29]. Also, I took care to avoid augmenting the validation set. The training and validation sets were already balanced. That means augmentation shouldn't be applied to the test dataset [29]. Otherwise, it can inflate accuracy artificially, returning irrelevant results. The goal here at the end is that a model trained on augmented data would be generally robust when tested on an untouched test dataset, except when some test-time augmentations are applied in order to get better results or by estimating aleatoric uncertainty [29].

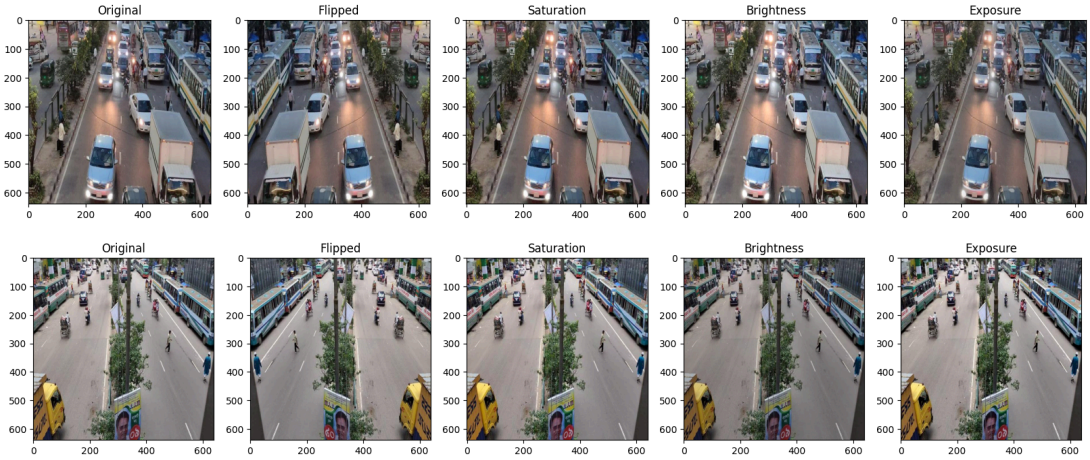

Figure 3.9: Images after Different Data Augmentation Techniques.

3.3 MODEL SELECTION

For this project, I chose recent state-of-the-art YOLO (You Only Look Once) [30] models that have high precision and speed in object detection tasks. Being single-stage detectors, the YOLO models [30] locate objects precisely under one pass; thus, they are appropriate for real-time applications such as traffic congestion detection.

3.3.1 OBJECT DETECTION

In this paper, I leverage state-of-the-art models for the object detection module: pre-trained YOLOv9 [21] and YOLOv10 [22]. The interesting thing about new YOLO models is related to their improved performance regarding detection of a wide range of objects within complex scenes. This is very essential in this piece of work since it has the goal of precisely identifying vehicles and pedestrians from an image of traffic.

- **YOLOv9**: This model can maintain a good balance between speed and accuracy. Its architecture is enhanced over its previous versions to provide high efficiency for real-time object detection tasks [21]. Therefore, since the good generalization across different datasets obtained by YOLOv9, it stands in its place to be used in our traffic images dataset as it is diverse.
- **YOLOv10**: As the latest member in the YOLO family, YOLOv10 has innovatively improved its performance both over accuracy and detection ability. Also, it might be highly reliable for models that identify small and densely packed objects in traffic scenes [22]. High precision and recall rates are required in our application for traffic congestion detection, thus being well-backed by the optimization done over the architecture and training techniques of YOLOv10.

3.3.2 CONGESTION REGION DETECTION

I shall make use of the output from the YOLO object detection model when tasking in identifying congested areas with the traffic images and with Eigen-CAM [23] for explainability. This has been chosen so as to not only have an accurate identification of the congested areas but also interpretability in results.

- **YOLOv10**: With a high mAP, the output of YOLOv10 is used in identifying objects like cars, bikes, and pedestrians. Because of the strength it has in detecting objects for any condition, it will work best for our purpose. Further, the bounding boxes given by YOLOv10 help in making an estimate of congestion levels based on object density and overlap.

- **Eigen-CAM:** I implemented Eigen-CAM to increase the detectability of the congestion regions. It is a gradient-free visualization method, visualizing principal components for the learned features/representations from convolutional layers, without modifying layers or requiring model retraining [23]. This approach visualizes areas in an image that most strongly activate a model's predictions, thus helping explain why certain regions are found as congested. This transparency is important for verifying the results produced by this model and allowing accountability in traffic management applications.

3.4 MODEL DETAILS

I will further explain in this section the models used in my thesis, including features, architecture, and the implementation of some of the key features. This subsection includes some figures and equations to clarify models effectively.

3.4.1 YOLOv9

While YOLOv9 puts many novel concepts in the limelight, Programmable Gradient Information and Generalized Efficient Layer Aggregation Network, to further make object detection tasks efficient and more accurate, Wang et al. ensure that this model handles deep network data loss through PGI by maintaining key features and reliable gradients during optimal training. On its part, GELAN maximizes the utilization of the parameters with high computational efficacy, thereby making YOLOv9 very adaptable with high performance across various applications (Wang, Yeh, and Liao, 2024).

YOLOv9 Key Components:

1. **The Information Bottleneck Principle:** It's a principle that lays better ground for describing the loss of information, which is done on data transformation incurred within a neural network [21]. It is measured by mutual information laid on the map between the original and transformed data [21].

   Information Bottleneck Equation,

   $$I(X, X) \geq I(X, f_{\theta}(X)) \geq I(X, g_{\phi}(f_{\theta}(X))) \quad (3.1)$$

   Here,

   I = Mutual Information

   X = Data

   f_{θ} and g_{ϕ} = Transformation Functions

During the traversal of data X through layers f_θ and g_ϕ of a deep neural network, some crucial information gets lost that is very important for the prediction [21].

2. **Reversible Functions:** Such functions allow transforming between data without loss of information and find an exact reconstruction of input data from network outputs [21].
   Reversible Function Equation,

$$X = \upsilon_\zeta(r_\psi(X)) \tag{3.2}$$

   Here,

   υ and r = Forward and Reverse Transformation

   X = Data

   These networks, using reversible functions in their layers, can preserve the complete information contained in the inputs and are thus more reliable for computing gradients for refinement [21].
3. **Programmable Gradient Information (PGI):** PGI involves inheritance from one main branch for inference, an additional reversible branch for computing gradients accurately, and multi-level auxiliary information [21]. It accrues deep supervision without extra inference overhead [21].
4. **Generalized Efficient Layer Aggregation Network (GELAN):** GELAN complements PGI, increasing the model's ability to handle data more effectively and learn from it, while folding in CSPNet gradient path planning as well as ELAN speed optimizations [21].

One of the variants already comes with YOLOv9e, and it drastically reduces parameters and computational demands over previous versions or competing models. However, high precision has been delivered in the application. For traffic congestion detection, it is essential to retain key information and generate reliable gradients for YOLOv9e. Advanced attention mechanisms and efficient backbone networks guarantee real-time and accurate vehicle and congestion pattern detection, which makes the model very suitable for any application in urban traffic management.

3.4.2 YOLOv10

YOLOv9 faces several critical challenges. First is the eliminability of non-maximum suppression (NMS), while the second concerns computationally redundant computations. YOLOv10 makes the assignment dual and eliminates the need for NMS at the inference stage. Large latency reduces while remaining competitive in performance (Wang et al., 2024).

Key Components in YOLOv10:

- **NMS-free training with consistent dual assignments:** This help avoids using NMS, which reduces the inference latency and thus improves the prediction efficiency [22]. It consolidates one-to-many and one-to-one assignment strategies to provide rich supervisory signals during training in one hand and enhance learning accuracy on the other [22].

  Model Architecture:

  - **Neck**: This neck is deliberately designed to gather all features from different scales and use PAN (Path Aggregation Network) to fuse multiscale features effectively.
  - **Backbone**: An enhanced version of CSPNet (Cross Stage Partial Network) to improve gradient flow without computation redundancy.
  - **One-to-Many Head**: Makes multiple predictions per object at train time, and these rich supervisory signals provide the training set.
  - **One-to-One Head:** Produces a single best prediction per object at inference, removing the need for NMS.

  Consistent Matching Metric: Another important module for the dual-label assignment strategy is the consistent matching metric to be computed when it comes to concordance evaluation between predictions and ground-truth instances [22]. The measurement considers a classification score and IoU between predicted and real bounding boxes, defined as:

$$m(\alpha, \beta) = s \times p^{\alpha} \times IoU(\hat{b}, b)^{\beta} \quad (3.3)$$

  Here,

  α, β are the hyperparameters that balance the impact of the semantic prediction task and the location regression task.

  $\hat{b}$ = Predicted Bounding Box

  b = Actual Bounding Box

  p = Classification Score

  s = Spatial Error

- **Holistic Efficiency-Accuracy Driven Model Design:** The depth of optimization done over every single model component guarantees minimal information loss and computational cost.

- **The expanded abilities of the model:** With further integration, large kernel convolutions and partial self-attention modules could enhance performance without adding much computational cost.

Given my thesis on traffic anomaly detection, for real-time applications, a dual-assignment strategy with efficient model design in YOLOv10L is required. High accuracy results at low computational costs present this model as an ideal choice for the detection and analysis of traffic congestion patterns in an urban environment.

3.4.3 EIGEN-CAM

This Eigen-CAM represents the chief components of the learned features from the convolutional layers, proving robust against classification mistakes by the fully connected layers in a CNN. It does not rely on backpropagation of gradients or class relevance scores, thereby being one of the versatile tools for visual explanations in neural networks. The main building block of the feature extraction network is the convolutional layers. At early stages, the convolutional layers learn lower level spatial features such as edges and corners. Coupled with the hierarchical structure, the convolutional layers learn higher levels of abstractions and possibly features that can produce semantics to at least a categorical level. On the other hand, the classification network is the fully-connected layers for flattening the learned features and drawing of the decision boundary for classifying the concepts (Muhammad and Yeasin, 2020).

3.5 HYPER-PARAMETER TUNING

The same hyperparameters were applied to YOLOv9 and YOLOv10 so that the evaluated performance could be consistent and comparable. The major hyperparameters include the number of epochs, batch size, learning rate, optimizer, weight decay, dropout rate, and label smoothing. These hyperparameters have been picked for experiments because of their possible significant impact on the performance of the model and the stability of training.

Classes:

The nine classes, for which these models were trained to detect, are as follows: Bike, Bus, Car, CNG, Cycle, Mini-Truck, People, Rickshaw, and Truck. This was the adopted classification scheme to completely envelop all kinds of vehicles or entities that move about in traffic circumstances in Dhaka.

Training Parameters:

- **Epochs**: For the YOLOv9 model, it consisted of 50 epochs of training, whereas for the YOLOv10 model, it was 120. This considers the differential complexities of these two models and their learning capabilities.
- **Batch Size**: For both the models, a batch size of 6 was trained so that at one end, the memory constraints would be harnessed and the efficiency of training on the other.
- **Learning Rate:** The initial learning rate was 1×10^{-3}, with a learning rate factor of 0.01. The learning rate plays an important role in developing how fast models converge and, at the same time, their stability.
- **Optimizer:** AdamW optimizer was used and it kept an independent adaptive learning rate for every subgradient. A default setting for AdamW is to take a learning rate of 7.69×10^{-4} and momentum of 0.9.

Regularization Parameters:

- **Weight Decay:** Weight decay was followed with 5×10^{-4} as the weight-decay parameter. This will avoid the overfitting caused by large weights that are penalized.
- **Dropout:** Dropout is configured at a rate of 0.0, which infers that while training, it does not implement any form of dropout regularization.
- **Label Smoothing:** A smoothing factor of 0.0 was used, which generally improves calibration and confers more robustness to the model.

Tuning Process:

- **Adjustment:** Usually carried out in iterations, this process tunes the hyper-parameters by checking the performance measures and training behavior. For instance, the learning rate has to be fine-tuned to compromise between the speed of convergence and stability.
- **Early Stopping**: I added a patience parameter so as to include early stopping. Training would stop in case no performance improvement had been recorded concerning the model's performance on the validation set through 20 epochs in a row.

This resulted in hyperparameter tuning that made it easy to identify the optimal settings through which the performance of both the YOLOv9 and YOLOv10 models could be maximized. These parameters ensured an effective learning of the training dataset and generalized to detect the traffic congestion regions with a high accuracy.

3.6 MODEL TRAINING

One important step for high performance in object detection tasks lies in the effective training of deep learning models. This thesis presents the application of the YOLOv9 to YOLOv10 models' detection of objects in road scenes to traffic congestion region detection using adequate training, with a very clearly described process of training these models to achieve maximal detection capabilities. It offers a process for model training, including the loading of a pre-trained model, setting of training parameters, and leveraging resources better from a GPU.

3.6.1 LOADING PRE-TRAINED MODELS

First, it loaded pre-trained YOLO models. Pre-trained weights aid in jumpstarting training both in speed and accuracy by knowledge transfer from large datasets.

Pre-trained models loaded were:

1. “YOLOv9e” for the YOLOv9 model
2. “YOLOv10l” for the YOLOv10 model

For these models, loading was done using the “ultralytics” library [31], designed for fast implementation of the YOLO architecture.

3.6.2 TRAINING IMPLEMENTATION

The steps put in use to train a model are therefore stated as:

Training the Model: Using the prepared dataset and previously specified hyperparameters, the models were trained. The training followed where the model learned from what was fed into it, to identify/classify the images using a loss function that would optimize through backpropagation.

Validation: Validation set (20% of the Dataset) was used to validate the performance of the models to track their generalization abilities. Validation metrics like precision, recall, and mean Average Precision are recorded for tracking the progress of models.

Fine-tuning: In case the performance was poor, it was fine-tuned based on validation performance by adjusting the learning rate and other hyper-parameters. So, in this way, this optimum configuration of the models could be reached through an iterative process.

GPU Utilization: Training models for object detection in deep learning is computationally expensive. For this instant research, we used an “NVIDIA P100 GPU” [32] for faster training. This can utilize the processing capabilities of this one with a large batch of images at high-resolution settings. This will significantly increase the speed at which the model trains.

Model Checkpoints and Resuming Training: Model checkpoints were saved periodically to improve the efficiency of the Training process. This would ensure resuming from the last saved

state in case of generalized and unanticipated interruptions of the training process. I set “resume=True”, load the saved weights, and continue training. This helped avoid losing of progress and thus optimize computational resource and time usage.

Model Exporting: Then I made sure to save the model after all the epochs were completed. I exported the model in "onnx" format. The 'best.pt', and 'last.pt' weights of both models were saved including all results.

3.7 “TCRD” FRAMEWORK IMPLEMENTATION

The Traffic Congestion Region Detection (TCRD) framework is a holistic solution designed to provide an analysis of the detection of congested region levels in an urban traffic scenario using state-of-the-art object detection models. This section gives the implementation of the TCRD framework, including pseudocodes.

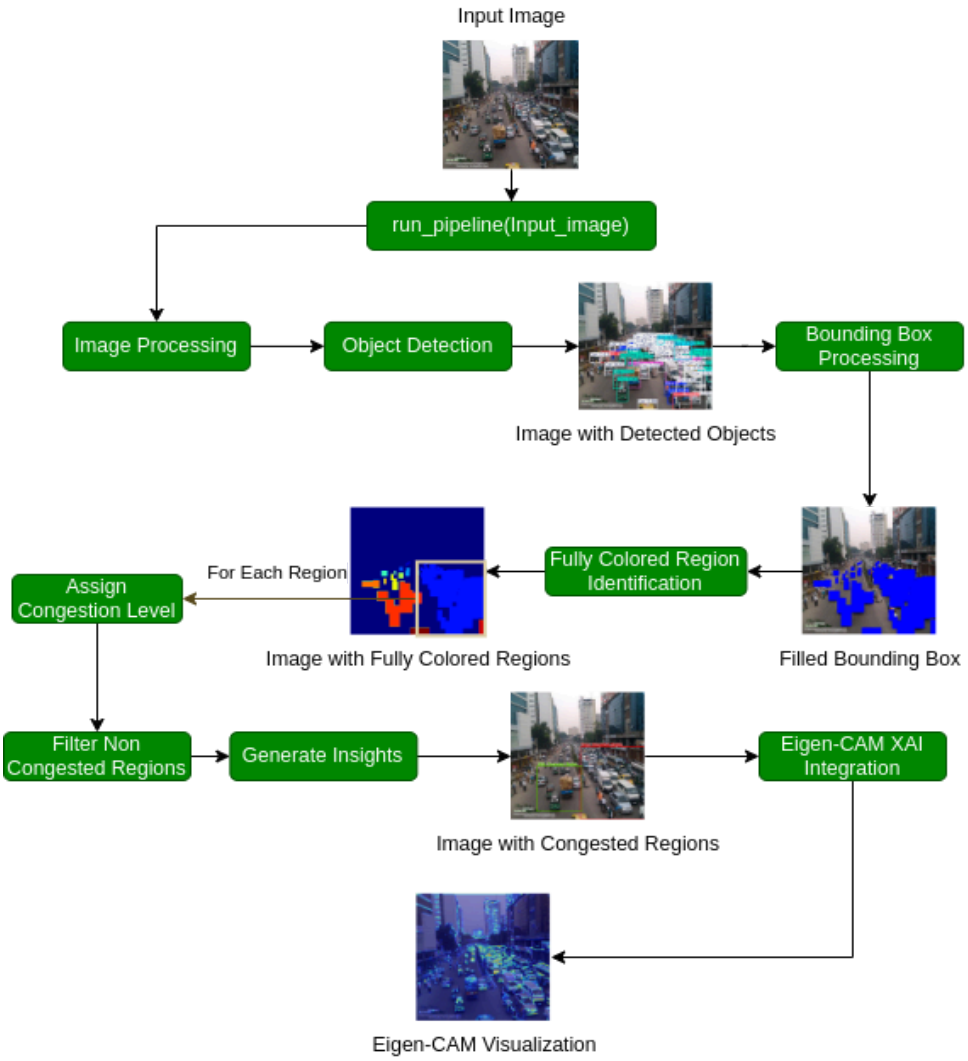

Figure 3.10: Traffic Congestion Region Detection Framework.

3.7.1 IMAGE PROCESSING

First of all, I load and then display the input image to ensure this is the format to be used for processing. In the "display_image(image_path)" function, I read the image using OpenCV's "cv2.imread()" and turned it into RGB color space using "cv2.cvtColor()" [33]. Finally, I have plotted this image using "matplotlib" [34].

Image Processing Algorithm

```
1. function display_image(image_path):
2.     image = read image from image_path
3.     convert image from BGR to RGB
4.     display image using matplotlib
```

3.7.2 OBJECT DETECTION

Since both the train and validation mAP of YOLOv10 were greater compared to that of YOLOV9, I took this version of YOLOv10 in this framework. So the saved model of YOLOv10 was loaded to detect objects in the input image. The detected objects are vehicles and people and probably the most important entities while analyzing traffic congestion. I then used the "model.predict()" function to detect objects in the test input image. It takes the path to the image, source; a list of indices of classes to detect; and the threshold of the confidence, conf. This function also resizes the image to 640 × 640 pixels using imgsz. It optionally saves the detected image, the results of detection in YOLO format, and confidence scores if specified. The function filters detections using the confidence score and then returns these results, including the bounding box coordinates, class indices, and confidence scores.

Object Detection Algorithm

```
1. function model_predict(input_image_path):
2.     results = model.predict(
3.         source=input_image_path,
4.         classes=[0, 1, 2, 3, 4, 5, 6, 7, 8],
5.         conf=0.30,
6.         imgsz=(640, 640),
7.         save=True,
8.         save_txt=True,
9.         save_conf=True,
10.        exist_ok=True
11.    )
12.    return results
```


3.7.3 PROCESSING BOUNDING BOXES

The detected bounding boxes are then filled with a specified color, blue, to highlight the detected objects. It is supposed to identify wholly colored regions in the image, indicating congestion. I applied the "cv2.rectangle()" function through the "fill_bounding_boxes(image_path, annotations)" function to draw filled rectangles in the image for each detected object based on the annotation text file.

Bounding Box Processing Algorithm

```
1. function fill_bounding_boxes(image_path, annotations):
2.     image = read image from image_path
3.     for each annotation in annotations:
4.         draw filled rectangle on image at annotation coordinates
5.     display image with filled bounding boxes
6.     return image
```

3.7.4 FULLY COLORED REGION IDENTIFICATION

I need to identify fully colored regions within the image. This shall be done by creating a mask for the blue-filled bounding boxes and finding connected components in the mask. Each connected component would form a different fully colored region in the image. This step helps in segmenting the image into different regions where bounding boxes are present. The result of this is important for the congestion level analysis. "identify_colored_regions(image)" returns coloured region labels together with the total coloured regions found.

Fully Colored Region Identification Algorithm

```
1. function identify_colored_regions(image):
2.     mask = create_mask_for_colored_boxes(image)
3.     num_labels, labels_im = find_connected_components(mask)
4.     return labels_im, num_labels
```

3.7.5 ASSIGNMENT OF CONGESTION LEVEL

This step will count the number of bounding boxes lying inside each fully colored region, and the congestions will be assigned according to the defined thresholding values. The congestion will be defined in the "assign_congestionlevels(labels_im, annotations)" function as :

1. Heavy Congested Region: More than 20 boxes
2. Moderate Congested Region: 15 to 20 boxes
3. Mild Congested Region: 10 to 15 boxes
4. Light Congestion Zone: 7-10 boxes

Congestion Level Assignment Algorithm

```
1. function assign_congestion_levels(labels_im, annotations):
2.     initialize congestion_levels as empty list
3.
4.     for each region_label from 1 to max(labels_im):
5.         create mask for region_label
6.         set num_boxes to 0
7.
8.         for each annotation:
9.             if annotation overlaps with mask:
10.                increment num_boxes
11.
12.        determine congestion_level based on num_boxes:
13.            if num_boxes > 20: level = 0, label = "Heavy"
14.            else if 15 <= num_boxes <= 20: level = 1, label = "Moderate"
15.            else if 10 <= num_boxes <= 15: level = 2, label = "Mild"
16.            else if 7 <= num_boxes <= 10: level = 3, label = "Light"
17.            else: continue
18.
19.        append (region_label, num_boxes, label, level) to congestion_levels
20. return congestion_levels
```

3.7.6 VISUALISATION AND INSIGHTS

The congestion regions detected were then visualized and insights were developed relating to the types and quantities of vehicles in every region.

In the "draw_congested_regions(image_path, labels_im, congestion_levels)" function, I have drawn bounding boxes around the congested regions labeled by their level of congestion.

The "generate_insights(congestion_levels, annotations, labels_im)" function takes every congested region and computes the distribution of different types of vehicles, then prints insights pertaining to them. This insight explains what kind of vehicle type sort is responsible for the traffic anomaly.

Insights Generation Algorithm

```
1. function draw_congested_regions(image_path, labels_im, congestion_levels):
2.     image = read image from image_path
3.     for each congestion_level in congestion_levels:
4.         draw bounding box and label on image
5.     display image with congested regions
6. function generate_insights(congestion_levels, annotations, labels_im):
7.     for each congestion_level in congestion_levels:
8.         calculate vehicle counts in region
9.         print insights
```


3.7.7 INTEGRATING XAI

I have implemented a function for how to visualize the Class Activation Map (CAM), which points out the detected objects in the image and highlights those regions of the image that contributed the most toward its detection, for explainable AI insights. In the "visualize_cam(input_image_path)" function, I applied the method of "Eigen-CAM" on a target layer "model.model.model[1]" to generate a CAM for the input image and then display it with Matplotlib. I noticed that this layer returned the clearest visualization of how different regions of an image contribute to a model's decision through iteration; it is therefore more interpretable and effective in analyzing object detection results as CAMs.

Eigen-CAM Visualization Algorithm

```
1. function visualize_cam(input_image_path):
2.     target_layers = [model.model.model[1]]
3.     img = read and preprocess image
4.     cam = EigenCAM(model, target_layers, task='od')
5.     grayscale_cam = cam(rgb_img)[0, :, :]
6.     cam_image = show_cam_on_image(img, grayscale_cam, use_rgb=True)
7.     display cam_image
```

3.7.8 INFERENCING PROCESS

The following is the fully end-to-end implementation of the "run_pipeline" function. This pipeline runs right from object detection to visualization, in other words, predicts objects of interest in the input image by feeding the set of detections into the model and then displaying the results and saving the annotations. It then loads the annotation, processes the image to fill boxes with color, and finally uses connected component analysis to find out which regions have become completely filled with color. Thereafter, it uses a bounding box density application to create congestion levels against these regions. This function call is responsible for drawing the congestion level text on an image, saving updated annotations, and generating further insights relating to the vehicle count by congestion level. Finally, this visualization on class activation maps gives insights into how the model came to a certain decision.

Object Detection Algorithm

```
1. function run_pipeline(input_image_path):
2.     results = model.predict(
3.         source=input_image_path,
4.         classes=[0, 1, 2, 3, 4, 5, 6, 7, 8],
5.         conf=0.30,
6.         imgsz=(640, 640),
```



```
        save=True,
        save_txt=True,
        save_conf=True,
        exist_ok=True
    )
    display_image(output_image_path)
    annotations = load_annotations(annotation_file_path)
    filled_image = fill_bounding_boxes(input_image_path, annotations)
    labels_im, num_labels = identify_colored_regions(filled_image)
    congestion_levels = assign_congestion_levels(labels_im, annotations)
    draw_congested_regions(input_image_path, labels_im, congestion_levels)
    save_annotations(congestion_levels, annotation_file_path)
    generate_insights(congestion_levels, annotations, labels_im)
    visualize_cam(input_image_path)
```

3.8 EVALUATION METHODS

To make our approach an unbiased and accurate exercise, we apply several metrics and techniques in our framework. In order to evaluate models YOLOv9 and YOLOv10, the following metrics will be used.

Precision Curve: It shows the model's precision across different confidence thresholds. It indicates the proportion that positive identifications are correct and measures how accurate the predictions are [35].

Recall Curve: It passes the recall of the model through various confidence thresholds, which gives the proportion of the actual positives correctly identified. Recall, thus, answers how good the model is at finding all positives [35].

F1-score Curve: Harmonic mean of precision and recall at different thresholds. This curve offers a single metric balancing both precision and recall [35].

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.6}$$

Here,

TP (True Positive): Correctly predicted positive instances.

TN (True Negative): Correctly predicted negative instances.

FP (False Positive): Incorrectly predicted positive instances.

FN (False Negative): Incorrectly predicted negative instances.

Precision-Recall Curve: A graph that exhibits the amount of trade-off between precision and recall for different thresholds. Through this graph, one can learn how the model works at various levels of confidence [35].

Confusion Matrix Normalized: It contains information about how the classification part of the detection works. It will have information regarding the proportion of correct and incorrect predictions, normalized over all instances.

IoU: Intersection over union—this measures how much our predicted boundary overlaps with the ground truth. In other words, it is a real object boundary [35].

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.7}$$

mAP50: Thus, this metric measures the mean Average Precision at an intersection over Union threshold of 0.5. It looks at how well the detected bounding boxes overlap with the ground truth boxes [35].

mAP50-90: This is used to check mean average precision over a range of IoU thresholds from 0.5 to 0.95, with a step size of 0.05. This gives an all-rounded assessment of model performance [35].

$$mAP_{50} = \frac{1}{N}\sum_{i=1}^{N} AP_{50}(i) \tag{3.8}$$

$$mAP_{50-95} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{10}\sum_{j=0}^{9} AP_{0.5+0.05j}(i) \tag{3.9}$$

Here,

N = Number of Classes

AP_k = Average Precision of Class K

Curves of Loss and Performance on Train and Val sets across epochs:

- **Box Loss:** Error in predicting coordinates of the bounding box.
- **Class Loss:** Error in predicting Class Labels.
- **DFL Loss:** Distribution Focal Loss that enhances localization accuracy.
- **Precision, Recall, mAP50, mAP50:95:** Performance metrics evaluated on the validation set for all epochs.

Eigen-CAM Insights: I make use of Eigen-CAM to show the regions in the image on which the model focuses during detection and provide insights into the decision-making capability of the model.

Finally, I infer the developed framework for congestion region detection on a test set example image to visualize and validate the detection of congested regions. This qualitative evaluation supplements the quantitative metrics and provides a holistic view of model performance.

3.9 SUMMARY

Therefore, the methodology that is followed here includes preparing a huge dataset of images, pre-processing them and augmenting them, before training the YOLOv9 and YOLOv10 models using the traffic congestion dataset. Thereafter, I computed such metrics as mAP50, mAP50-95, precision, recall, and F1-score for the purpose of performance evaluation. As a matter of fact, I used Eigen-CAM for visual explanation of the model's prediction, which makes it more explainable. Finally, the implementation of this framework and its efficiency in detecting congestion regions, and further classification with respect to the number of detected objects, were tested to gain full knowledge regarding the traffic congestion in an urban city like Dhaka.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 INTRODUCTION

In this chapter, I will report the results and analysis of my traffic congestion detection framework. First of all, I compare the best performances of YOLOv9 and YOLOv10 object detection models by table. Then, I will provide a normalized confusion matrix for the classification performance of the models. Training and validation graphs are given where loss and performance metrics across training epochs are plotted. I have also provided how the test image would look if passed through this TCRD framework, along with visualizations using Eigen-CAM for highlighting areas of congestion. I then discuss the results and insights that can be obtained about the detected traffic anomalies in Dhaka, which were mainly aimed at shedding light on the effectiveness and real-world use cases of the model.

4.2 RESULT ANALYSIS

4.2.1 OBJECT DETECTION RESULTS

I measured a number of key metrics for the performances of YOLOv9 and YOLOv10, including mAP50, mAP50-95, precision, recall, and loss values. The best performance for both models on these metrics is summarized in Table 4.1.

Table 4.1: Evaluation Metrics on Validation Set

Evaluation Metrics	YOLOv9	YOLOv10
Epoch	50	120
mAP50	0.7641	0.9382
mAP50-95	0.5085	0.7050
Precision	0.6674	0.9401
Recall	0.7644	0.9188
Train/Box Loss	1.0455	0.342
Train/Cls Loss	0.5659	0.2131
Train/Dfl Loss	1.0136	0.2868

Val/Box Loss	1.1861	0.3086
Val/Cls Loss	0.6325	0.1847
Val/Dfl Loss	1.1048	0.3043

Model YOLOv10 performed much better for all of the evaluation metrics than YOLOv9, where their corresponding mAP50 values are 0.9382 and 0.7641, respectively. Similar to the results for other methods, better precision and recall are indicated by its higher mAP50-95, whose value is 0.7050, while that for this latter is 0.5085. Its lower losses for training and validation point to improved efficiency of learning and model generalization. These results clearly indicate that on this context of traffic congestion, the YOLOv10 model was more effective at detecting objects.

4.2.2 OBJECT DETECTION - CONFUSION MATRIX

Realization of the confusion matrix for the YOLOv9 and YOLOv10 models indicates that there was a staggering improvement in YOLOv10 over YOLOv9. All classes are very well predicted in YOLOv9, but there is high misclassification into the "Cycle" category, with 37% being misclassified as "People," and high errors into the "Background" category. On all classes, YOLOv10 had better accuracy, significantly reducing their misclassification rates to 21% for "Cycle" and "Background". While this represented an improvement over the baseline, it continued to misclassify into the "Background" category, especially against "Truck" and "Cycle". In sum, better performance is expected, but more fine-tuning is needed for handling misclassification issues.

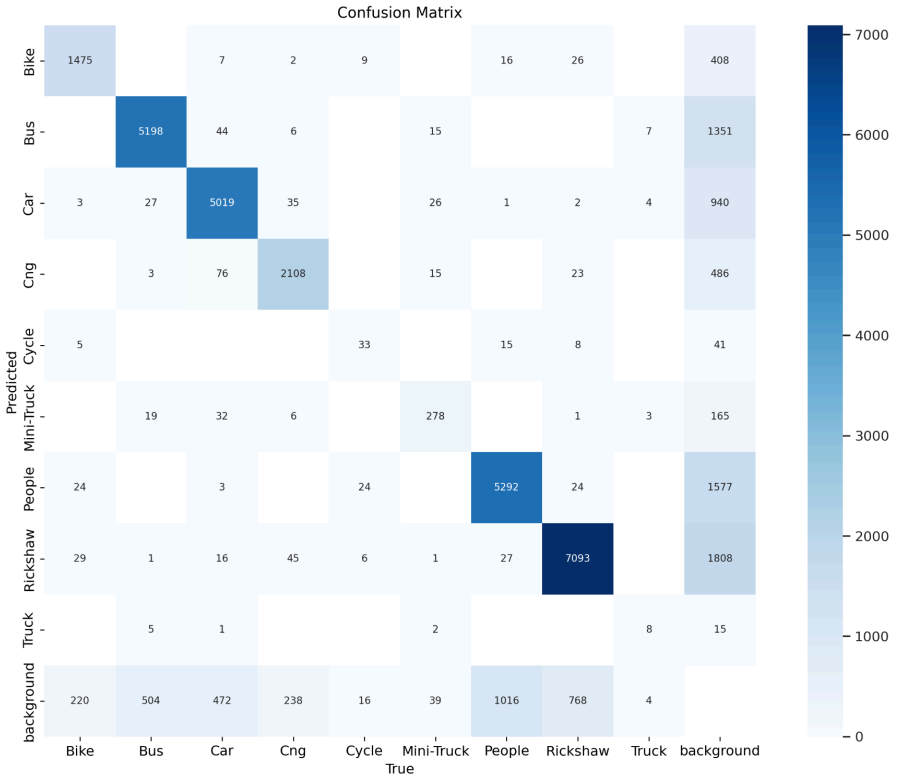

Figure 4.1: Confusion Matrix of YOLOv9.

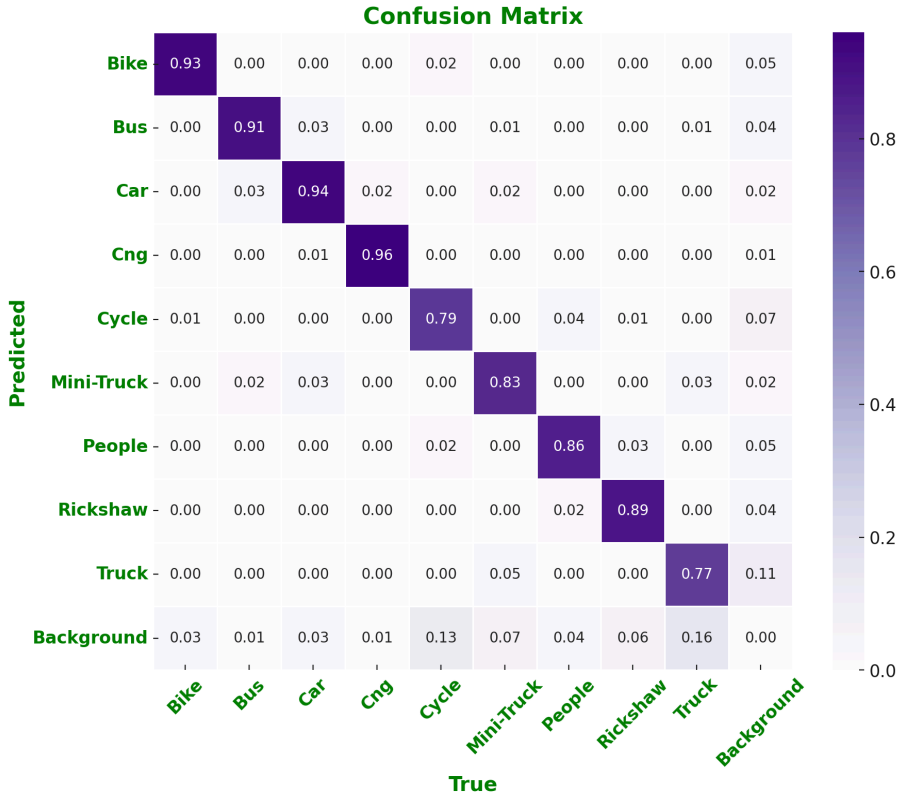

Figure 4.2: Confusion Matrix of YOLOv10.

4.2.3 OBJECT DETECTION - RESULT VISUALIZATION

The YOLOv9 visualization delivers information regarding important metrics and losses across training epochs. Specifically, the training losses for train/box_loss, train/cls_loss, and train/dfl_loss all show a smooth decrease, showing improved performance in the quality of bounding box predictions and classification, and on the distribution-focused loss measures. The metrics leading towards precision and recall initially go up and down but overall trend upwards, meaning improvement in precision and recall over time. Validation losses like val/box_loss and val/cls_loss first go down and eventually remain flat, whereas the val/dfl_loss rises at the very end, which may indicate overfitting. Both metrics/mAP50(B) and metrics/mAP50-95(B) graphs increase and are stabilized, hence improving Mean Average Precision.

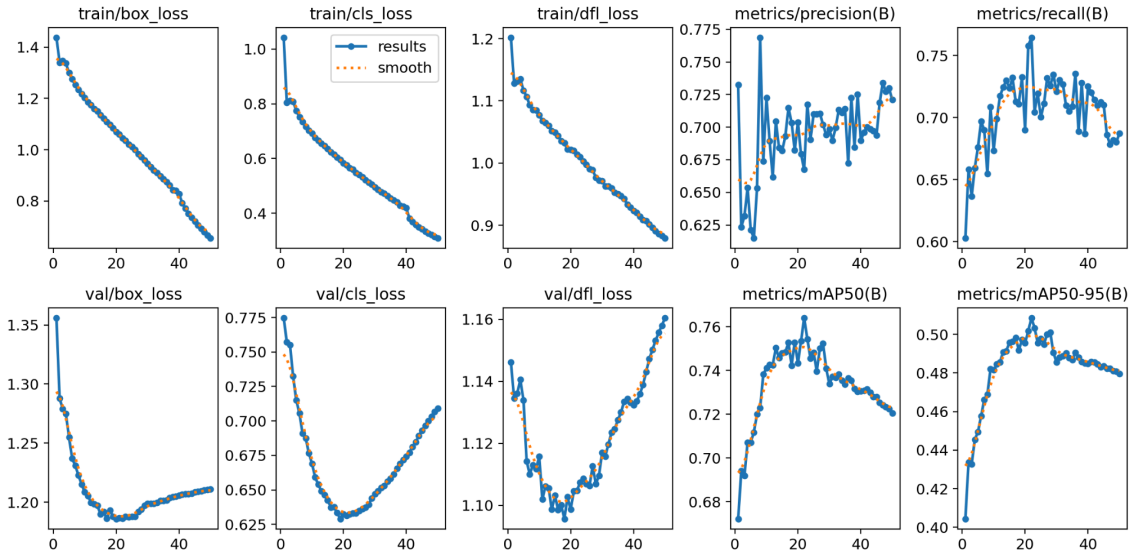

Figure 4.3: YOLOv9 Object Detection Training, Validation Loss and Performance Graph.

YOLOv10 visualization for 120 epochs shows continuous improvement in training and validation metrics. These training losses are compatibly decreasing, while precision and recall metrics are rising, thus portraying better performance. Other metrics, such as metrics/mAP50(B) and metrics/mAP50-95(B), also demonstrate an upward trend. The validation losses are steadily in a downward toggle; thus, good generalization toward unseen data was achieved. Overall, YOLOv10 outweighed YOLOv9 with superior performance, since it had more consistent and lower losses and steadily improving precision, recall, and mAP metrics.

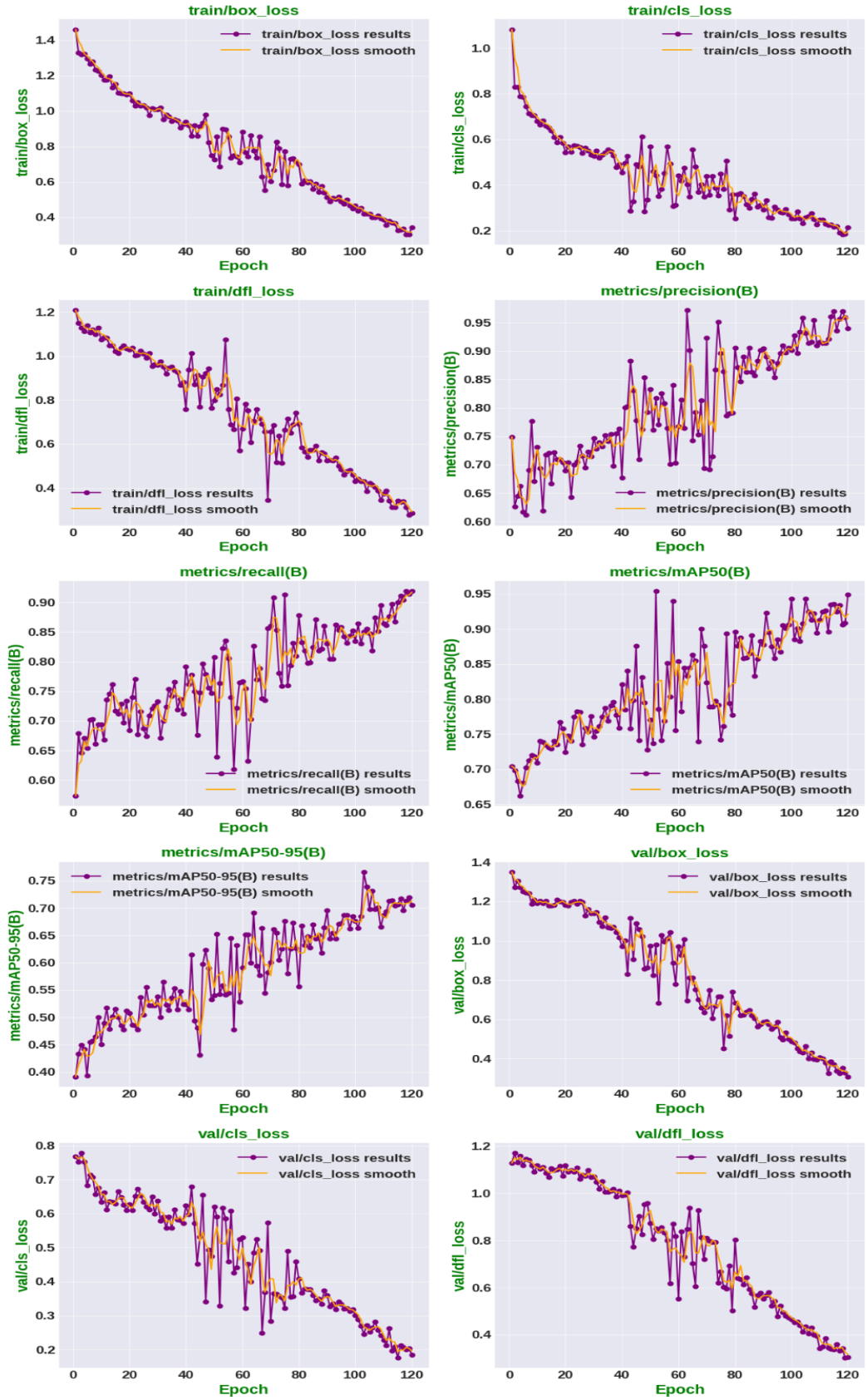

Figure 4.4: YOLOv10 Object Detection Training, Validation Loss and Performance Graph.

4.2.4 TRAFFIC CONGESTION REGION DETECTION

The following (Figure 4.5) is the result after processing the sample image of the test set through the TCRD framework. My trained YOLOv10 model was able to detect some objects within the image: 5 bikes, 4 buses, 42 cars, 14 CNG, and 2 mini-truck in just 2750.6ms.

```
image 1/1
/kaggle/working/Traffic-Congestion-Detection-1/test/images/85_jpg.rf.69d0d5f1b3f78cf19c4aa
d790169a214.jpg: 640x640
5 Bikes, 4 Buss, 42 Cars, 14 Cngs, 2 Mini-Trucks, 2 Peoples, 2750.6ms
Speed: 9.6ms preprocess, 2750.6ms inference, 10.1ms postprocess per image at shape (1, 3,
640, 640)
Results saved to runs/detect/predict
1 label saved to runs/detect/predict/labels
```

Figure 4.5: Vehicles and People Detection Output.

Next, blue was set as the color of the bounding boxes so as to better visualize the filled areas (Figure 4.6).

Annotations: [(186, 494, 236, 599, 'Cng'), (236, 371, 259, 416, 'Cng'), (589, 465, 639, 557, 'Car'), (127, 421, 180, 469, 'Car'), (295, 427, 340, 485, 'Car'), (168, 457, 209, 535, 'Cng'), (514, 412, 585, 466, 'Car'), (260, 381, 284, 421, 'Cng'), (459, 315, 539, 389, 'Bus'), (400, 363, 439, 398, 'Car'), (55, 371, 109, 401, 'Car'), (235, 416, 296, 530, 'Mini-Truck'), (460, 383, 513, 430, 'Car'), (200, 374, 225, 415, 'Cng'), (341, 341, 371, 388, 'Car'), (380, 396, 417, 449, 'Car'), (417, 394, 488, 503, 'Mini-Truck'), (369, 361, 400, 412, 'Car'), (148, 318, 175, 346, 'Car'), (378, 284, 408, 311, 'Bus'), (388, 331, 413, 357, 'Car'), (230, 329, 252, 357, 'Car'), (474, 410, 511, 476, 'Car'), (504, 438, 608, 562, 'Car'), (375, 313, 398, 337, 'Car'), (263, 322, 282, 347, 'Car'), (437, 360, 474, 395, 'Car'), (452, 571, 537, 640, 'People'), (540, 370, 573, 407, 'Cng'), (285, 606, 378, 639, 'Car'), (471, 499, 583, 610, 'Car'), (491, 365, 520, 408, 'Cng'), (587, 410, 630, 465, 'Cng'), (411, 317, 438, 339, 'Car'), (354, 320, 376, 343, 'Car'), (429, 552, 455, 612, 'Bike'), (517, 387, 575, 431, 'Car'), (233, 301, 250, 320, 'Car'), (337, 329, 359, 355, 'Car'), (108, 366, 139, 405, 'People'), (612, 566, 639, 640, 'Car'), (396, 312, 413, 331, 'Car'), (316, 316, 326, 341, 'Car'), (367, 337, 398, 361, 'Car'), (438, 320, 460, 345, 'Car'), (360, 308, 376, 323, 'Car'), (414, 516, 437, 571, 'Bike'), (413, 299, 426, 315, 'Cng'), (543, 356, 573, 377, 'Cng'), (262, 282, 272, 294, 'Car'), (209, 318, 216, 340, 'Bike'), (434, 335, 460, 360, 'Car'), (422, 307, 442, 321, 'Car'), (202, 358, 221, 379, 'Cng'), (572, 351, 640, 444, 'Bus'), (202, 358, 221, 376, 'Cng'), (194, 336, 205, 366, 'Bike'), (506, 313, 520, 327, 'Cng'), (572, 350, 640, 446, 'Bus'), (160, 358, 172, 385, 'Bike'), (341, 309, 360, 329, 'Car'), (270, 278, 278, 287, 'Car'), (348, 282, 361, 298, 'Car'), (357, 273, 370, 285, 'Car'), (277, 271, 284, 283, 'Car'), (334, 303, 344, 318, 'Car'), (617, 440, 639, 498, 'Cng'), (353, 299, 367, 311, 'Car'), (266, 306, 283, 325, 'Car')]

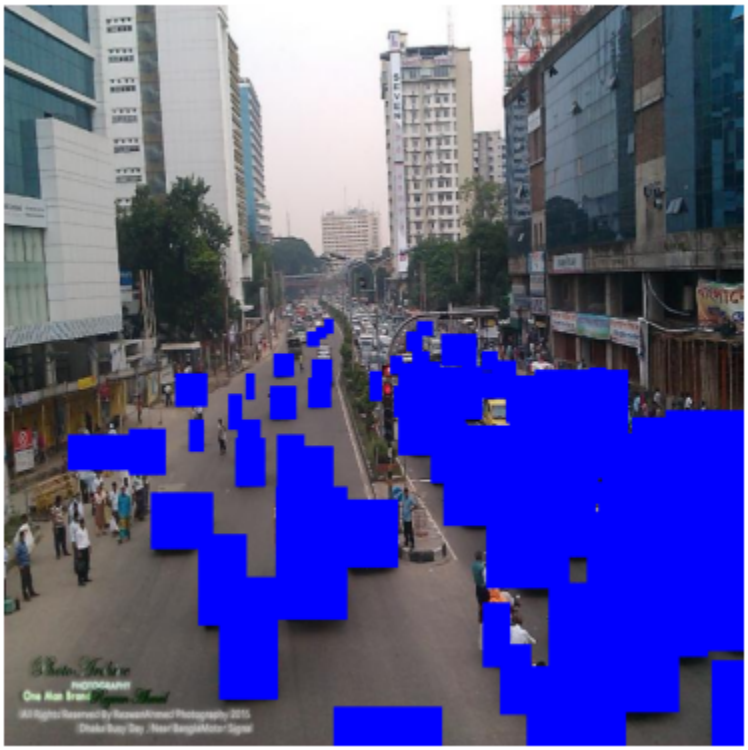

Figure 4.6: Blue Filled Bounding Boxes in the Image.

The frame was masked with various colors for complete local regions upon detection, and the coordinates with the corresponding number of boxes were shown (Figure 4.7).

Number of fully colored regions: 15
Region 1: 3 boxes
Region 2: 43 boxes
Region 3: 1 boxes
Region 4: 2 boxes
Region 5: 1 boxes
Region 6: 1 boxes
Region 7: 1 boxes
Region 8: 1 boxes
Region 9: 4 boxes
Region 10: 1 boxes
Region 11: 2 boxes
Region 12: 7 boxes
Region 13: 1 boxes
Region 14: 1 boxes
Congestion Levels: [(2, 43, 'Heavy Congested Region', 0), (12, 7, 'Light Congested Region', 3)]

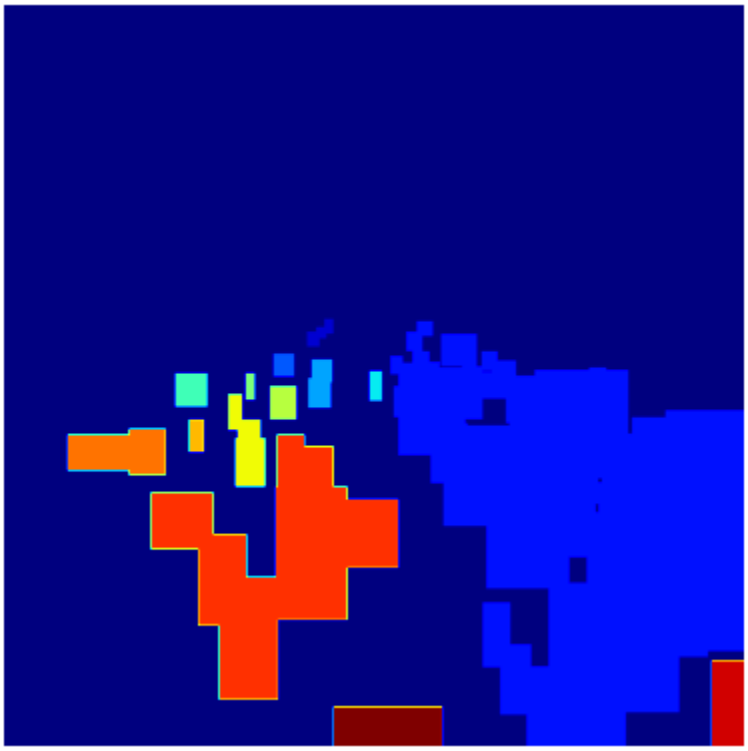

Figure 4.7: Fully Colored Region Masking.

The colored regions, hereby (Figure 4.8) totally identified by the image, amounted to 15. Among these, the most packed region was Region 2, for the breakup of the bounding boxes in this region that showed it to be 43, and was categorised in "Heavy Congestion Region". There were 2 Bikes, 4 Buses, 28 Cars, 7 CNGs, 1 Min i-Truck, and 1 Person. The other important region was Region 12, which was classified as a "Light Congested Region" because it had 7 bounding boxes: 2 Cars, 4 CNGs, and 1 Mini-Truck.

Heavy Congested Region with 43 boxes: {'Bike': 2, 'Bus': 4, 'Car': 28, 'Cng': 7, 'Cycle': 0, 'Mini-Truck': 1, 'People': 1, 'Rickshaw': 0, 'Truck': 0}
Light Congested Region with 7 boxes: {'Bike': 0, 'Bus': 0, 'Car': 2, 'Cng': 4, 'Cycle': 0, 'Mini-Truck': 1, 'People': 0, 'Rickshaw': 0, 'Truck': 0}

Figure 4.8: Traffic Image with Congested Regions and Categories.

In this instance, there could be a likelihood of congestion due to a large number of cars occupying space in Region 2. The observation creates the impression that there is to be road alternative infrastructure that has to be put in place so as to avoid congestion.

Indeed, the provided images depict the detected objects with bounding boxes, the colored filled boxes, and the fully and lastly colored and congested regions. In this paper, a full visual and numerical analysis of the results has been shown, emphasizing the potential capability of this model to accurately detect and classify congestion levels, hence giving important insights about traffic anomalies in Dhaka.

4.2.5 EXPLAINABLE AI DECISIONS

Eigen-CAM Analysis:

After applying Eigen-CAM on my trained YOLOv10 model's target layer "model.model.model[1]" and processing the same sample image, the results were:

```
0: 640x640 6 Bikes, 4 Buss, 46 Cars, 14 Cngs, 5 Mini-Trucks, 2 Peoples, 2450.0ms
Speed: 2.7ms preprocess, 2450.0ms inference, 0.4ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict
2 labels saved to runs/detect/predict/labels
```

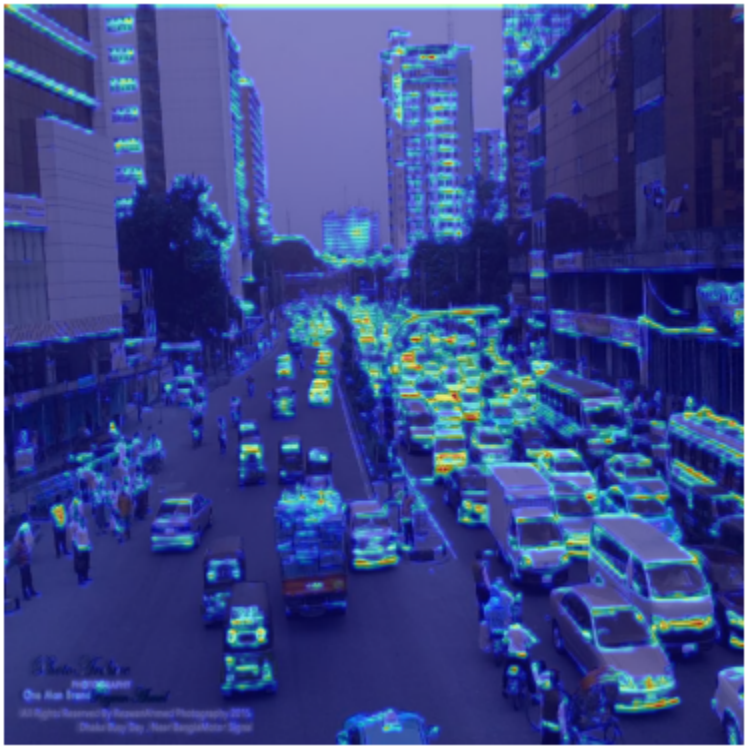

Figure 4.9: Eigen-CAM Analysis on Sample Image of Test Set.

Model's Decision and Explainability:

- **Object Detection Variations:**
    - **Original YOLOv10 Prediction (Figure 4.5):** 5 Bikes, 4 Buses, 42 Cars, 14 CNGs, 2 Mini-Trucks, 2 People
    - **Eigen-CAM Analysis:** 6 Bikes, 4 Buses, 46 Cars, 14 CNGs, 5 Mini-Trucks, 2 People

  It is observed in the Eigen-CAM analysis that there is an increase in the detected number of Bikes, Cars, and Mini-Trucks. This directly implies that the model, when visualized

through Eigen-CAM, recognizes more objects than the original prediction. The slight variations visible in the number of detected objects could have been because of the model's focus on the different regions lightened up by the Eigen-CAM.

- **Inference Time Improvement:** Eigen-CAM reduced the inference time from 2750.6 milliseconds to 2450.0 milliseconds, thus indicating that it optimizes any prediction by focusing on key image areas.
- **Insight Gained from Visualization:** The key image regions driving predictions are visually highlighted through Eigen-CAM and happen to be the same as the objects of bikes, cars, buses, CNGs, or mini-trucks.
- **Explainability:** Eigen-CAM increases model transparency since it elaborates on what drives detections—something very important in letting one understand model decisions for complex tasks such as traffic congestion detection.
- **Fine-Tune Potential:** Differences in object detection counts by Eigen-CAM and original predictions point to areas of model fine-tuning, that is, for improving the accuracy of detection for objects like Bikes, Cars, and Mini-Trucks.
- **Traffic Anomalies Insight:** Compared with original predictions, Eigen-CAM might become very useful in the understanding of the nature of traffic and designing effective traffic management based on the types of vehicles detected in the most congested regions.

Such insights can be used to fine-tune the model for better accuracy and build up better strategies for traffic anomaly management in an urban setup like Dhaka.

4.3 KEY INSIGHTS AND DISCUSSION

High-Density Vehicle Region: It is a heavily crowded region with 28 cars. The share of private cars is on the higher side, which contributes much to traffic congestion. Improvement in road infrastructure or alternate transportation arrangements may help in managing the flow.

Role of Public Transport: Region 2 also has 4 buses and 7 CNGs. While public transport is indispensable, its better scheduling, dedicated lanes, and efficient routing can reduce its contribution towards traffic congestion.

Impact of Vehicle Diversity: With a range of vehicles, such as bikes, mini-trucks, etc., there is a need to have adaptive techniques of management to accommodate varying sizes and speeds.

Light Congested Areas: Region 12 with fewer vehicles (2 cars, 4 CNGs, 1 mini-truck) would benefit from policies like improving traffic signals and lane discipline to further reduce congestion.

Spatial Distribution: Essentially, it gives region levels in 15 areas, which helps in prioritizing infrastructure development and traffic management in areas of varied congestion intensities.

Infrastructure Betterment: Granular levels of traffic congestion analysis indicate that there is a requirement for focused infrastructure betterment in the form of dedicated lanes or road expansions so that bottlenecks can be reduced.

Policy and Planning: Vehicle composition and congestion patterns have to be factored into effective traffic policy. A cut in private car usage, improvement in public transport, and encouragement of alternative modes of transport will help reduce traffic congestion.

Overall, it is through a step-by-step presentation of results that some hidden insights into this traffic management problem can be revealed. By looking at the details of each region, it may enable us to figure out why certain colored regions or, at worst, the fully colored region is congested. This includes the number and type of objects in that region. For example, in Region 2, there is a high density of congestion; this could be attributed to a large number of cars taking up space. In that respect, the inference of the observation may indicate that alternative road infrastructure would help alleviate the challenge of congestion. In addition, it would be possible to establish whether the congested parts include rickshaws, which are slow vehicles and normally cause accidents, with a view to traffic management.

4.4 RESULT COMPARISON

Compared to earlier works (Table 4.2), using YOLOv10 with Eigen-CAM will allow for better performance in traffic congestion detection with improved interpretability and transparency, attaining a higher mAP50 of 0.94. This progress delineates valuable insights about the traffic anomalies that earlier methods were unable to address.

Table 4.2: Comparison with Previous Literature.

Related Works	Method	mAP50	Main Contribution
Sundas et al. (2023)	YOLO, YOLOv3, YOLOv4, Faster R-CNN, Cascade R-CNN	0.93	Utilized deep learning-based UAVs with various detectors for traffic congestion control but lacked interpretability.
Bindu et al. (2023)	CNN	0.93	Achieved high accuracy in vehicle detection from surveillance video, for traffic accident and congestion detection.
Tan & Kieu (2023)	YOLOv5 + Deep SORT	0.93	Implemented vehicle tracking with high mAP but did not address low-light conditions.
Proposed Method (2024)	YOLOv9 and YOLOv10 with Eigen-CAM in TCRD Framework	0.94	Achieved competitive mAP50 with enhanced interpretability through Eigen-CAM, addressing both congestion region detection, unlocking insights to traffic anomalies and model transparency.

4.5 SUMMARY

This framework of TCRD with YOLOv10 and Eigen-CAM provided the best performance in traffic congestion detection, reaching a notable mAP50 of 0.94. Unique insights were made into the exact identification of areas that are highly congested and the applicability of Eigen-CAM in highlighting the major congestion areas for the optimization of object detection and inference time. This framework makes a detailed inspection of vehicle types and densities, hence leading to actionable insights in targeted urban traffic management improvement in Dhaka.n traffic congestion prediction, making these outputs meaningful for practical applications.

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

In summary, I have developed a complete approach to the detection and analysis of traffic congestion in Dhaka by using deep learning models and explainable AI techniques. Rigorous experimentation results show the effectiveness of the framework developed in identifying and classifying traffic-congested regions, including objects detected within these areas and the analysis of their distributions. Appending both YOLOv9 and YOLOv10, I have attained a huge increase in the performance of detection precision to around 0.94 mAP50 and speed of inference. In this light, the application of the Eigen-CAM method has increased the insight into the very process of model decision-making, thereby increasing transparency and accountability within this exercise. My findings really underlined the importance of depth in traffic analysis in urban planning and traffic management to bring out important patterns in suggesting infrastructure improvements for decongesting the area. This work provides a fresh multi-stage framework in the literature where advanced object detection gets seamlessly combined with explainable AI, resulting in more effective and informative solutions to be devised for traffic management.

5.2 LIMITATIONS AND FUTURE WORK

Although the multi-stage traffic anomaly analysis framework proposed by me achieves good recognition accuracy, it still has some disadvantages, such as the use of image data provided by static images, which may not fully reflect the dynamic aspects of real traffic flow in practice. Models may also be sensitive to different environmental conditions not sampled in the dataset (e.g., catastrophic weather). There are many types of traffic anomalies in Dhaka city that should be detected as well in order to improve the traffic system like Vehicles on Sidewalks, Wrong Way Driving, Restricted vehicles, Road Distresses, etc.

In future work, the potential extension of the dataset to include real-time traffic data could construct a dynamic evaluation of the resource effects, making the models more adaptable to different traffic scenarios and even environmental variations. It is also important to investigate other traffic anomalies like wrong-way driving, vehicles on the sidewalk, as well as road distress. In some cases, we may need to add some other factors like time and map data. We possibly need some critical advancements in models which are probably graph neural networks.

5.3 CONTRIBUTION

The primary contribution of my approach is to develop a novel multi-stage Traffic Anomaly Analysis framework displaying the integration of Advanced Object Detection and Explainable AI techniques. I mainly focused on the datasets of targeted volumes from congestion-prone zones of Dhaka, such as Jatrabari, Shekertek, Dhanmondi, Mirpur, and New Market, by preprocessing and augmenting the dataset. Then, I developed a framework that utilizes YOLOv9, and YOLOv10 for highly accurate (0.94 mAP50) object detection, and Eigen-CAM for throwing light on the detailed visual explanation in identifying traffic congestion. The obtained data from this framework gives data-driven information on potential inputs: the number of distinct vehicles and pedestrians, hence indicating the detection of other unseen anomalies such as illegal parking, privately owned car parking space-occupying, and accident-prone vehicles like autorickshaws and rickshaws. This paradigm shift answers the bellow for transparent AI in city rendering and drives drastic progress in the field of traffic anomaly detection and management.

5.4 IMPLICATION

The developed Traffic Congestion Region Detection (TCRD) framework, through the integration of explainable AI, will lead to a strong support to the government sector for effective traffic management. Better traffic monitoring, better decision-making, and better policy implementation through accurate detection and explicit explanations of the traffic anomalies like congestion can lead to minimized congestion with a better flow of traffic and better urban planning of Dhaka and such urban areas. The use of such advanced techniques will bring about appropriate resource allocations and timely interventions, hence boosting the quality of urban life.

REFERENCES

[1] Mustafa, K. (2023). 'Why exactly is Dhaka the slowest city in the world?', The Daily Star. Available At: https://www.thedailystar.net/opinion/views/news/why-exactly-dhaka-the-slowest-city-the-world-3436751.

[2] Abu Afsarul Haider (2018). 'Traffic jam: The ugly side of Dhaka's development', The Daily Star. Available At: https://www.thedailystar.net/opinion/society/traffic-jam-the-ugly-side-dhakas-development-1575355.

[3] Borgalli, R.A. and Surve, S. (2023). 'Learning Framework for Real-World Facial Emotion Recognition', *AI-Enabled Social Robotics in Human Care Services*. https://www.igi-global.com/chapter/learning-framework-for-real-world-facial-emotion-recognition/322516.

[4] Coussement, K., Abedin, M.Z., Kraus, M., Maldonado, S. and Topuz, K. (2024). 'Explainable AI for enhanced decision-making', *Decision Support Systems,* [online] p.114276. Available at: https://doi.org/10.1016/j.dss.2024.114276.

[5] Tribune Desk (2023). 'Traffic Index: Dhaka has 5th worst traffic in world', Dhaka Tribune. Available at: https://www.dhakatribune.com/bangladesh/dhaka/308347/traffic-index-dhaka-has-5th-worst-traffic-in.

[6] Rahman, Md.M., Jamil, A.R.M. and Nower, N. (2023), 'Uncertainty-aware traffic prediction using attention-based deep hybrid network with Bayesian inference', *International Journal of Advanced Computer Science and Applications (IJACSA)*. Available at: http://dx.doi.org/10.14569/IJACSA.2023.01406132.

[7] Rahman, Md.M. and Nower, N. (2024), 'Inferring traffic patterns of Dhaka City: A spatio-temporal analysis over a year', *Research Square*. Available at: https://doi.org/10.21203/rs.3.rs-3827938/v1.

[8] Iftikhar, S., Asim, M., Zhang, Z., Ammar Muthanna, Chen, J., ElAffendi, M., Sedik, A. and Abd, A.A. (2023). 'Target Detection and Recognition for Traffic Congestion in Smart Cities Using Deep Learning-Enabled UAVs: A Review and Analysis', 13(6), pp.3995–3995. Available at: https://doi.org/10.3390/app13063995.

[9] G. Bindu Madhavi, A. Durga Bhavani, Y. Sowmya Reddy, Kiran, A., N. Thulasi Chitra and Shaker, C. (2023). 'Traffic Congestion Detection from Surveillance Videos using Deep Learning', Available at: https://doi.org/10.1109/ic2e357697.2023.10262545.

[10] Dang Minh Tan and Kieu, L.-M. (2023). 'TRAMON: An automated traffic monitoring system for high density, mixed and lane-free traffic', *IATSS Research*, 47(4), pp.468–481. Available at: https://doi.org/10.1016/j.iatssr.2023.10.001.

[11] Zunair, H., Khan, S. and Hamza, A. (2024). 'RSUD20K: A DATASET FOR ROAD SCENE UNDERSTANDING IN AUTONOMOUS DRIVING', Available at: https://arxiv.org/pdf/2401.07322.

[12] Fuad, M., Sifath, M., Amin, A., Ahmed, N., Hasan, M. and Ishraque, I. (2022). 'Traffic Congestion Prediction using Deep Convolutional Neural Networks: A Color-coding Approach', *IEEE Copyright Notice.* Available at: https://arxiv.org/pdf/2209.07943.

[13] Hossain, I. and Nower, N. (2022). 'Traffic Data Collection and Visualization Tool for Knowledge Discovery Using Google Maps', *International Journal of Software Innovation*, 10(1), pp.1–12. Available at: https://doi.org/10.4018/ijsi.293270.

[14] Ahad, K., Yasmeen, D. and Khan, R. (2021). 'Real Time Traffic Congestion Analyzer of Dhaka City', Available at: https://ist.edu.bd/wp-content/uploads/2021/09/Real-Time-Traffic-Congestion-Analyzer-of-Dhaka-City.pdf.

[15] Kurniawan, J., Syahra, S.G.S., Dewa, C.K. and Afiahayati (2018). 'Traffic Congestion Detection: Learning from CCTV Monitoring Images using Convolutional Neural Network', *Procedia Computer Science*, 144, pp.291–297. Available at: https://doi.org/10.1016/j.procs.2018.10.530.

[16] Bawaneh, M. and Simon, V. (2022). 'Novel traffic congestion detection algorithms for smart city applications', *Concurrency and Computation: Practice and Experience*, 35(5). Available at: https://doi.org/10.1002/cpe.7563.

[17] Nidhal, A., Ngah, U.K. and Ismail, W. (2014). 'Real time traffic congestion detection system', *[online] IEEE Xplore*. Available at: https://doi.org/10.1109/ICIAS.2014.6869538.

[18] Rahman, M., Shuvo, M., Zaber, M. and Ali, A. (2018). 'Traffic Pattern Analysis from GPS Data: A Case Study of Dhaka City', *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. Available at: https://doi.org/10.1109/CONECCT.2018.8482371.

[19] Chakraborty, P., Adu-Gyamfi, Y.O., Poddar, S., Ahsani, V., Sharma, A. and Sarkar, S. (2018). 'Traffic Congestion Detection from Camera Images using Deep Convolution Neural Networks', *Transportation Research Record: Journal of the Transportation Research Board*, 2672(45), pp.222–231. Available at: https://doi.org/10.1177/0361198118777631.

[20] Khan, S.W., Hafeez, Q., Khalid, M.I., Alroobaea, R., Hussain, S., Iqbal, J., Almotiri, J. and Ullah, S.S. (2022). 'Anomaly Detection in Traffic Surveillance Videos Using Deep Learning', *Sensors*, [online] 22(17), p.6563. Available at: https://doi.org/10.3390/s22176563.

[21] Wang, C.-Y., Yeh, I-Hau. and Liao, H.-Y.M. (2024). 'YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information', *arXiv (Cornell University)*. Available at: https://doi.org/10.48550/arxiv.2402.13616.

[22] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J. and Ding, G. (2024). 'YOLOv10: Real-Time End-to-End Object Detection', doi: https://doi.org/10.48550/arXiv.2405.14458.

[23] Muhammad, M.B. and Yeasin, M. (2020). 'Eigen-CAM: Class Activation Map using Principal Components', *2020 International Joint Conference on Neural Networks (IJCNN)*, [online] pp.1–7. Available at: https://doi.org/10.1109/IJCNN48605.2020.9206626.

[24] Dwyer, B. (2020). 'When should I auto-orient my images?', *Roboflow Blog*. Available at: https://blog.roboflow.com/exif-auto-orientation/.

[25] Villegas, I.D., Camargo, J.R. and Perdomo Ch., C.A. (2021). 'Recognition and Characteristics EEG Signals for Flight Control of a Drone', *IFAC-PapersOnLine*, [online] 54(4), pp.50–55. Available at: https://doi.org/10.1016/j.ifacol.2021.10.009.

[26] Islam, M.M., Rashid, M.R., Das, A. and Hasan, Md.R. (2024). 'Bangladeshi Traffic Flow Dataset', *[online] Mendeley Data*. Available at: https://doi.org/10.17632/h8bfgtdp2r.3.

[27] ASM Shihavuddin and Rifat, M. (2020). 'DhakaAI', *Harvard Dataverse*. Available at: https://doi.org/10.7910/dvn/porexf.

[28] Waskom, M., (2014). 'seaborn: statistical data visualization', *seaborn 0.9.0 documentation*. [online] Pydata.org. Available at: https://seaborn.pydata.org/.

[29] Barreto, S. (2022). 'Data Augmentation', *Baeldung on Computer Science*. Available at: https://www.baeldung.com/cs/ml-data-augmentation.

[30] KALRA, K. (2023). 'YOLO (You Only Look Onc', [online] Medium. Available at: https://medium.com/@khwabkalra1/yolo-you-only-look-onc-523b01ec4f4d.

[31] Jocher, G., Chaurasia, A. and Qiu, J. (2023). 'YOLOv8 by Ultralytics', [online] GitHub. Available at: https://github.com/ultralytics/ultralytics.

[32] G.G., S.M., (2016). 'TESLA P100 PCIe GPU ACCELERATOR. NVIDIA', Available at: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/solutions/resources/documents1/NV-tesla-p100-pcie-PB-08248-001-v01.pdf

[33] nkmk (2019). 'Convert BGR and RGB with Python, OpenCV (cvtColor)', *note.nkmk.me*. Available at: https://note.nkmk.me/en/python-opencv-bgr-rgb-cvtcolor/.

[34] Matplotlib (2012). 'Matplotlib: Python plotting', *Matplotlib 3.1.1 documentation*. Available at: https://matplotlib.org/.

[35] Hui, J. (2019). 'mAP (mean Average Precision) for Object Detection', Available at: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173