TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PASHCHIMANCHAL CAMPUS, POKHARA

Lamachaur, Pokhara-16

FINAL PROJECT REPORT ON

"Video-Based Vehicle Classification"

Submitted by

Aashish Katila	**[PAS076BCT001]**
Ashish Acharya	**[PAS076BCT007]**
Nitesh Devkota	**[PAS076BCT019]**
Saurab Bhattarai	**[PAS076BCT037]**

Submitted to

Department of Electronics and Computer Engineering

Falgun, 2080

COPYRIGHT

ACKNOWLEDGEMENT

We would like to acknowledge with much appreciation towards the Department of Electronics and Computer Engineering, Pashchimanchal Campus, Pokhara and Er. Khem Raj Koirala, Head of Department for providing us the opportunity to work on this Major project as part of our syllabus. Our sincere gratitude extends to each contributor who played a crucial role in this endeavor. Special appreciation is reserved for our supervisor, Mr. Ramesh Thapa, whose guidance, consistent oversight, and invaluable information significantly contributed to the project's progress. His constant feedback and suggestions throughout the development of this project helped us to face and tackle the challenges and complications we encountered while working on this project. We are also indebted to our friends for their unwavering support and motivation throughout the project's development. Lastly, we express our thanks to all individuals directly or indirectly involved in the project.

Sincerely,
Aashish Katila
Ashish Acharya
Nitesh Devkota
Saurab Bhattarai

ABSTRACT

The escalating complexities stemming from burgeoning vehicular traffic necessitate innovative solutions for efficient management and surveillance. This study is centered on devising and assessing a machine learning-based model for video-based vehicle identification and classification. The principal aim is to elevate automation and precision in traffic surveillance operations. Utilizing the You Only Look Once (YOLO) algorithm, the model undergoes comprehensive training and validation utilizing diverse vehicular scenarios. The meticulously curated and annotated dataset facilitates the model's adeptness in real-time recognition of various vehicle types in video streams. Performance evaluation encompasses crucial metrics such as accuracy, precision, recall, and processing speed. The research delves into potential applications spanning traffic monitoring, parking management, and law enforcement. Furthermore, the model's resilience in tackling challenging environmental conditions is thoroughly examined. The insights garnered from this study can serve as valuable resources for further research and analysis by other researchers and practitioners. The results reveal that the model achieves an overall precision of 0.861, an overall recall of 0.763, and an overall mAP50 of 0.828. These commendable scores signify the efficacy and reliability of the model in vehicle detection within images.

Keywords: *Machine Learning, Vehicle Identification, Video Analytics, You Only Look Once (YOLO), Traffic Surveillance*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GPU : Graphical Processing Unit

AI : Artificial Intelligence

YOLO: You Only Look Once

RNN : Recurrent Neural Networks

CNN : Convolutional Neural Network

CVAT : Computer Vision Annotation Tool

CHAPTER 1 : INTRODUCTION

1.1 Background

The escalating growth of vehicular traffic in urban areas necessitates advanced solutions for effective traffic management and surveillance. Traditional methods of vehicle identification and classification are often manual, labor-intensive, and prone to errors. In response to these challenges, machine learning algorithms have gained prominence for their potential to automate these processes. The project focuses on utilizing the You Only Look Once (YOLO) machine learning algorithm for CCTV-video-based vehicle identification and classification. YOLO is renowned for its real-time object detection capabilities, making it a compelling choice for applications requiring swift and accurate analysis of visual data. Existing state-of-art vehicle detection methods are not so robust in harsh environmental conditions and night-time conditions and exhibit below par level of performance. This project seeks to address these limitations by leveraging the efficiency of YOLO. The objective is to develop a model that can accurately identify and classify diverse vehicle types, such as cars, trucks, and motorcycles, within video streams in different harsh environment conditions. To achieve this, a carefully curated dataset comprising various vehicular scenarios is used for model training and validation. This dataset is crucial for enhancing the model's ability to generalize and perform effectively across different environmental conditions. The study further aims to evaluate the model's performance against real-world video datasets, considering key metrics like accuracy, precision, recall, and processing speed. The outcomes of this project are expected to contribute to the advancement of intelligent transportation systems, enabling more efficient traffic monitoring and law enforcement. Moreover, the exploration of the model's robustness under challenging conditions, such as varying lighting and weather, adds practical value to its potential applications.

1.2 Problem Statement

The escalating urbanization has led to a surge in vehicular traffic, exposing the inadequacies of traditional manual methods for vehicle identification and classification. These approaches are labor-intensive, error-prone, and lack the agility required for dynamic traffic scenarios. Automated systems, though providing some relief, often fall short in terms of real-time processing and accuracy. This study addresses these challenges by employing the You Only Look Once (YOLO) machine learning algorithm, aiming to develop a model capable of accurately identifying and classifying various vehicles within video streams in real time. Challenges include optimizing YOLO for diverse scenarios, ensuring seamless handling of different vehicle types, and evaluating performance in real-world conditions. This research contributes insights into potential applications in traffic surveillance, parking management, and law enforcement. (References: Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection (Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.)

1.3 Objectives

- To implement and evaluate a real-time vehicle identification and classification model using the YOLO machine learning algorithm for efficient traffic surveillance.

1.4 Application

Main applications of our project are:

- Help in Emergency Services by providing the traffic free route.
- Help to manage the Traffic System of the whole city via Smart Traffic Management

CHAPTER 2 : LITERATURE REVIEW

In the late 1990s, the convergence of computer graphics and computer vision ushered in a transformative era marked by groundbreaking techniques such as image-based rendering, image morphing, and panoramic image stitching. These innovations revolutionized how visual data could be processed and analyzed, laying the groundwork for the advancement of various applications, particularly in the domain of object detection and classification. The burgeoning interest in object detection stemmed from the recognition of its pivotal role in addressing pressing real-world challenges, such as traffic management and surveillance.
With the advent of video surveillance systems, there emerged a ripe opportunity to leverage computer vision techniques for more efficient and accurate vehicle identification and classification. However, while the foundational research in object detection had laid a solid groundwork, there remained a pressing need to bridge the gap between theoretical advancements and practical applicability in real-world settings. To this end, researchers embarked on a quest to refine existing methodologies and develop novel approaches tailored to specific use cases and environmental conditions.

Central to these endeavors were advancements in deep learning architectures, particularly convolutional neural networks (CNNs), which emerged as a cornerstone technology in the realm of object detection. One seminal contribution came from Ross Girshick and his collaborators, who introduced the region-based convolutional neural network (RCNN) framework. This pioneering approach utilized a two-stage process, wherein candidate object regions were first proposed using techniques like selective search, followed by classification and bounding box regression using CNNs. Subsequent refinements, including Fast RCNN and Faster RCNN, aimed to enhance the efficiency and accuracy of object detection by streamlining the processing pipeline and incorporating insights from feature extraction and region proposal generation. Moreover, the evolution of object detection architectures saw the emergence of single-stage detectors like You Only Look Once (YOLO) and Single Shot Detector (SSD), which offered faster inference speeds albeit with a trade-off in precision. These advancements underscored the ongoing pursuit of striking a delicate balance between speed and accuracy, a fundamental consideration in real-world deployment scenarios.

In parallel, researchers delved into the realm of instance segmentation, seeking to delineate individual objects within an image with pixel-level precision. This led to the development of sophisticated models such as Mask RCNN, which not only detected objects but also generated high-quality segmentation masks for each instance, thus paving the way for more nuanced analysis and understanding of visual data. Beyond object detection, researchers also grappled

with the challenges posed by adverse weather conditions and low-light environments, which often hampered traditional surveillance systems.

Innovations such as fog detection algorithms and night-time vehicle detection methodologies sought to mitigate these challenges, enabling more robust and reliable performance across diverse environmental conditions. Furthermore, the creation of benchmark datasets, such as the expressway dataset and the DAWN dataset, provided researchers with standardized benchmarks for evaluating the efficacy of their algorithms under real-world conditions. These datasets, comprising diverse scenarios and environmental factors, served as invaluable resources for benchmarking performance and fostering collaboration within the research community. In summary, the intersection of computer graphics, computer vision, and deep learning has catalyzed a paradigm shift in object detection and surveillance systems. From foundational research to practical applications, the journey towards more robust, efficient, and versatile systems continues to unfold, driven by the relentless pursuit of innovation and the quest for real-world impact.

CHAPTER 3 : METHODOLOGY

3.1 Tools Used

For the project, the following libraries, frameworks and tools are used:

- Jupyter Notebook:

  Jupyter Notebook is a popular open-source web application for interactive computing. It enables users to create and share documents containing live code, visualizations, and explanatory text.

- Google Colab:

  Google Colab is a cloud-based platform that offers free access to Jupyter Notebooks with GPU support. It allows collaborative coding and execution of Python code in a browser environment.

- PyTorch:

  PyTorch is an open-source machine learning framework developed by Facebook's AI Research lab, offering dynamic computation graphs. It's widely used for tasks like deep learning and natural language processing due to its flexibility and ease of use.

- TensorFlow:

  TensorFlow is an open-source machine learning framework developed by Google. It's renowned for its scalability and efficiency in building and deploying machine learning models.

- Numpy:

  Numpy is a fundamental package for numerical computing in Python, offering powerful array manipulation capabilities. It provides essential tools for working with large datasets and performing mathematical operations efficiently.

- Pandas:

  Pandas is a Python library for data manipulation and analysis, providing easy-to-use data structures and functions. It's widely used for tasks like data cleaning, exploration, and transformation in data science and analytics projects.

- Matplotlib:

  Matplotlib is a Python library for creating static, interactive, and animated visualizations in a variety of formats. It's extensively used for data visualization tasks in fields such as data science, engineering, and academia.

- OpenCV:

  OpenCV is an open-source computer vision and machine learning software library, offering extensive functionality for image and video processing. It's widely used for tasks like object detection, facial recognition, and image enhancement in diverse applications ranging from robotics to healthcare.

3.2 Data Collection

The research centers around a multi-camera system tailored for tracking a central point. The cameras employed in this investigation record videos at a rate of 24 frames per second, boasting a resolution of 2304×1296. However, the recorded video stream is vulnerable to imperfections, including blurring, hardware glitches, and hindrances like spider webs. Given the diverse weather conditions at the research location, data frames were gathered during both rainy and sunny seasons to address distinct environmental challenges.

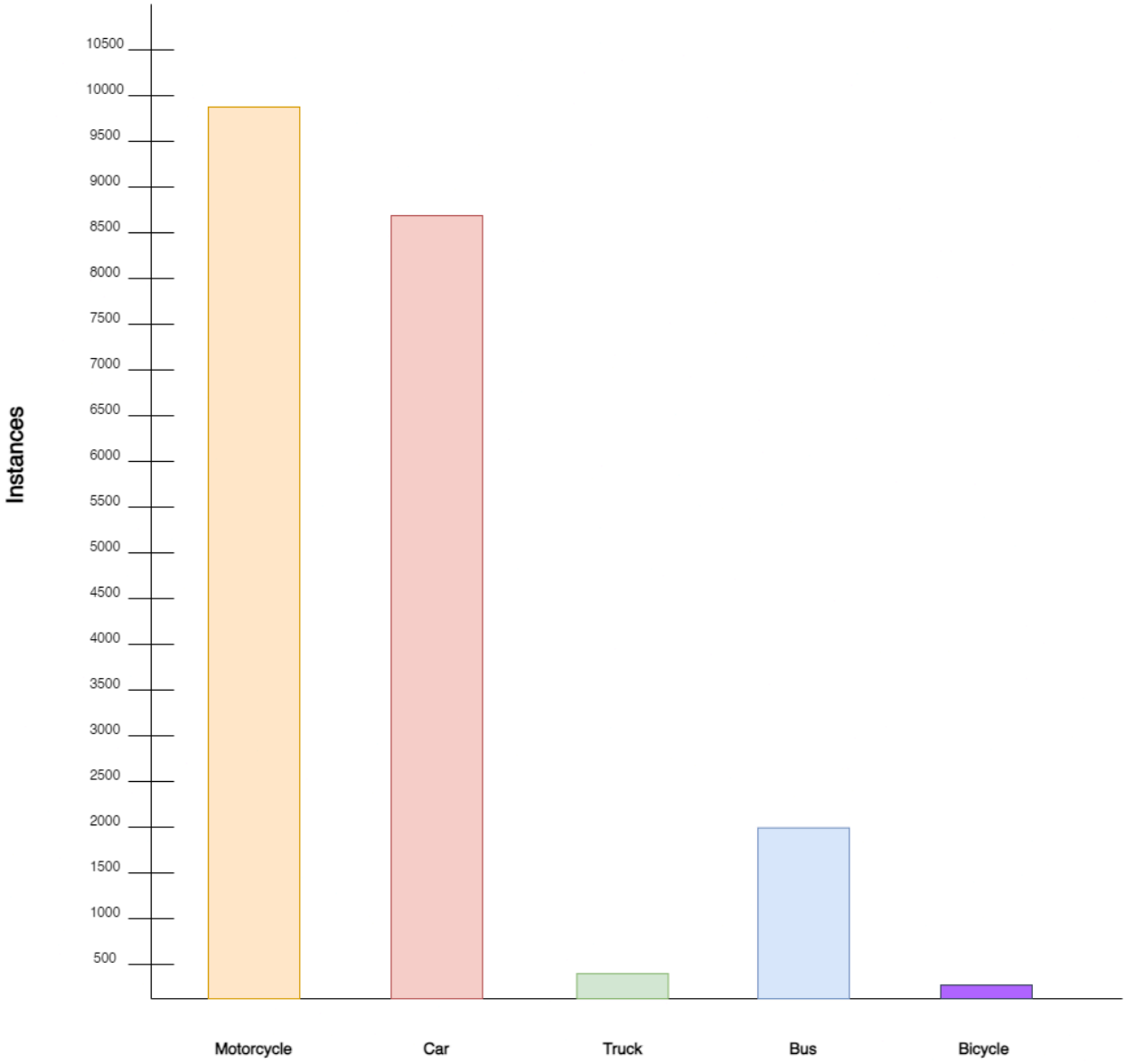

Figure 1: Bar Diagram for No.of Instances

Rather than annotating lengthy continuous video sequences, the focus was shifted towards brief clips encompassing various traffic scenarios to circumvent time-consuming intricacies. Two primary conditions were taken into account for the study: unfavorable weather conditions (rainy) and daylight-related lighting conditions (sunny). It was noted that the annotation of vehicles could be executed with a high level of confidence in both rainy and sunny atmospheres. To ensure the resilience of the dataset, instances from peak daytime conditions were also incorporated. Peak time conditions, acknowledged for their computational intricacies and the abundance of vehicle instances within a single frame, were deemed essential for the vehicle detection model.

The gathered dataset was categorized into three segments: Rainy, Sunny, and Nighttime. The Rainy set encompasses video clips recorded during the rainy season, the Sunny set encompasses diverse daytime lighting conditions, and the Nighttime set comprises traffic video clips recorded during the night. Specific time intervals were not accentuated, facilitating a more comprehensive representation of instances across day and night. Instances between 9 P.M. and 6 A.M. intentionally received less focus due to the scarcity of vehicles during this period and the challenges associated with confidently annotating vehicles captured in the camera frame.

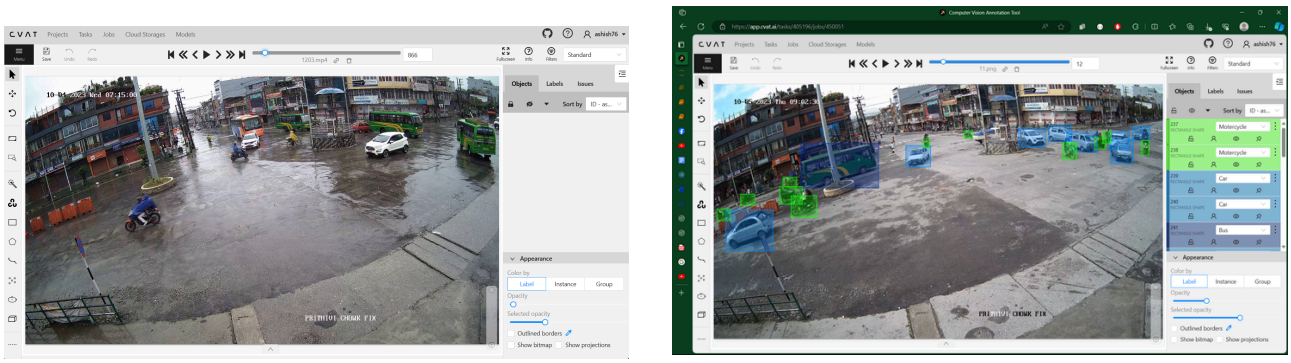

Figure 2: CVAT UI

3.3 Vehicle Annotation

We marked 21204 instances across 1200+ frames using rectangles in the Computer Vision Annotation Tool (CVAT), which served as the platform for all annotation tasks. Choosing CVAT over other annotation tools such as LabelIng, VGG annotator tool, and COCO annotator tool was based on its robustness in data annotation and deployment. CVAT, a docker-based annotation tool, stood out with its advanced features like interpolation.

The CVAT User Interface (UI) was utilized for annotation tasks, where vehicles were categorized into five types: Car, Truck, Bus, Motorcycle, and Bicycle. Our custom dataset analysis involved 9947 Motorcycle instances, 8645 Cars, 344 Trucks, 2013 Buses, and 255 Bicycles. The dataset's performance depended on the annotation rules applied:

Small Target: Recognizing that objects appear less distinct as they move farther away, small items were meticulously labeled, with bounding boxes adjusted to potential compression from a distant perspective.

Occlusion: The CVAT annotation tool featured a specialized function to address occlusions in the scene. This capability allowed us to position the bounding box of an occluded object behind the bounding box of the occluding object.

Special Samples: The dataset included a few special samples with category ambiguity, such as Tractor and Dozer. While present in the dataset, these samples were not labeled under the standard classes used in this particular dataset.

Figure 3: Representing Image Frame at day condition

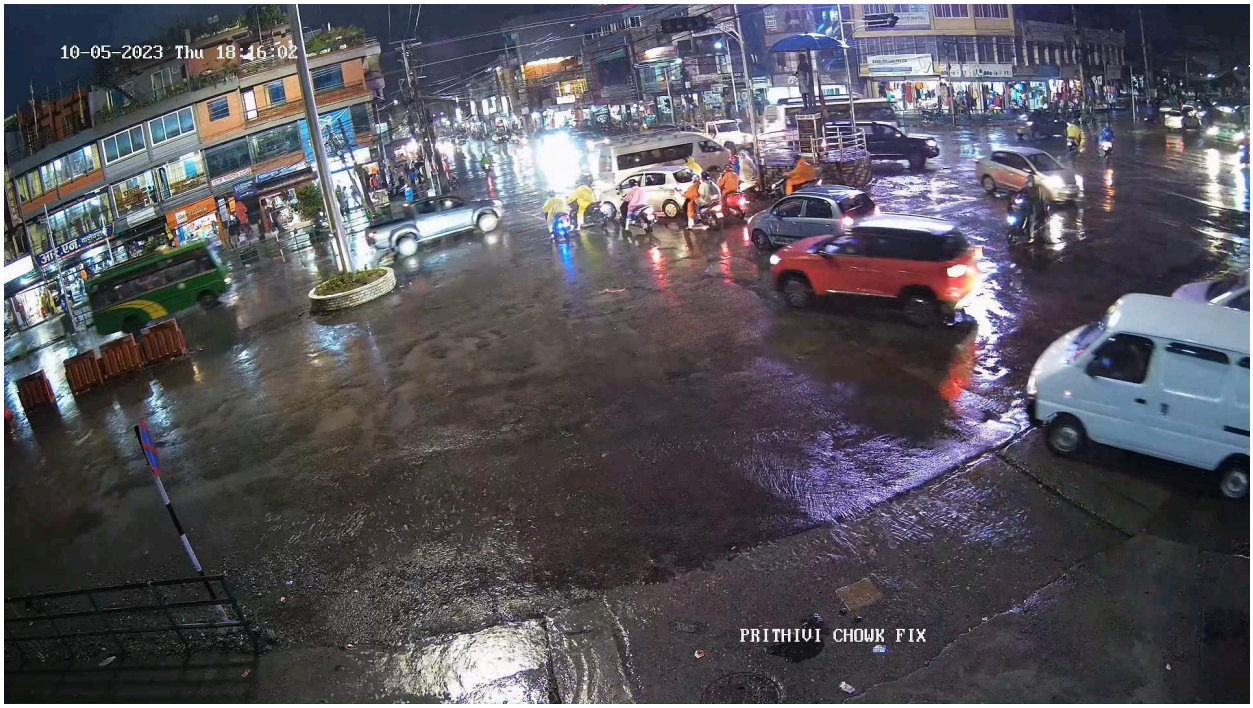

Figure 4 : Representing Image Frame at Night Time

Figure 5 : Representing Image Frame at Rainy time

3.4 Model Generation

The process of model generation involves the creation and development of a system that can effectively identify and classify vehicles using the YOLO algorithm. This encompasses the training and optimization of the model to ensure its accuracy and efficiency in real-world scenarios. The generated model is designed to capture and analyze diverse vehicular situations, contributing to its overall effectiveness in applications such as traffic surveillance and management. This phase is critical for producing a robust and reliable model capable of making accurate predictions in dynamic environments

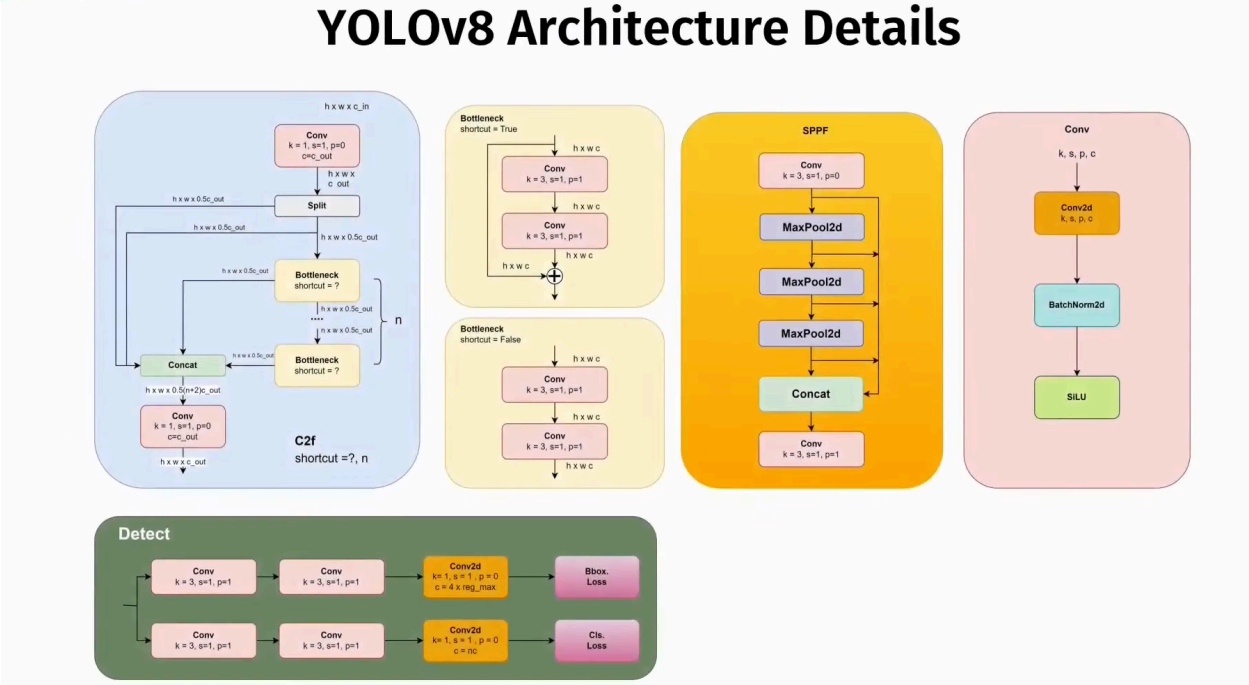

Figure 6 : YOLOv8 Architecture Details

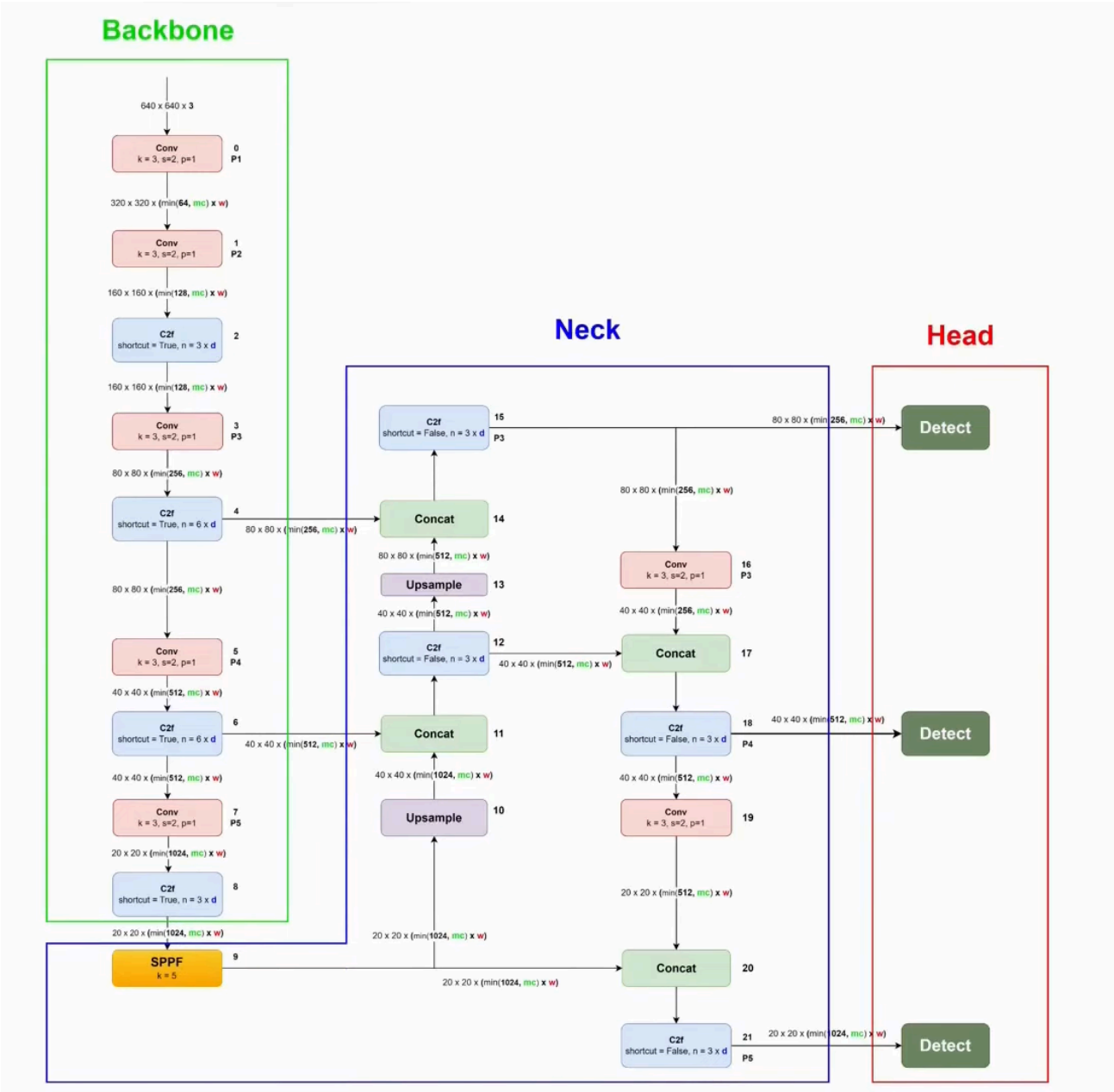

Figure 7: YOLOv8 Complete Architecture

3.4.1 Main Blocks:

The YOLO architecture comprises three pivotal blocks: Backbone, Neck and Head and everything will occur within these blocks. The function of each block is described below.

Backbone:

Purpose: Serving as the feature extractor, the backbone is tasked with extracting meaningful information from the input data.
Actions:
- Initially captures basic patterns like edges and textures in its early layers.
- Features multiple scales of representation, progressively abstracting information.
- Provides a comprehensive hierarchical representation of the input, aiding in subsequent analysis.

Neck:

Purpose: Acting as a connector between the backbone and the head, the neck performs operations for feature fusion and contextual integration.
Actions:
- Aggregates feature maps from different stages of the backbone to construct feature pyramids.
- Integrates contextual information to enhance detection accuracy by considering broader scene context.
- Reduces spatial resolution and dimensionality to streamline computation, thereby enhancing speed albeit potentially at the cost of model quality.

Head:

Purpose: Serving as the final stage of the network, the head is responsible for producing outputs such as bounding boxes and confidence scores for object detection.
Actions:
- Generates bounding boxes corresponding to potential objects within the image.
- Assigns confidence scores to each bounding box, indicating the likelihood of an object's presence.
- Categorizes objects within bounding boxes based on their respective classes.

Convolution:

In the YOLO architecture, a local feature analysis approach is employed rather than processing the entire image at once. This strategy aims to reduce computational complexity, facilitating real-time object detection. The algorithm relies on convolutions to extract feature maps.

Convolution, a fundamental mathematical operation, merges two functions to generate a third. In computer vision and signal processing, convolution is frequently utilized to apply filters to images or signals, emphasizing specific patterns. Within convolutional neural networks (CNNs), convolutions play a crucial role in feature extraction from inputs like images. They are characterized by parameters such as Kernels (K), Strides (s), and paddings (p).

In essence, convolutional operations in YOLO involve applying filters to input data, enabling the extraction of essential features for subsequent analysis. This process aids in identifying relevant patterns within images, ultimately contributing to the accurate detection of object

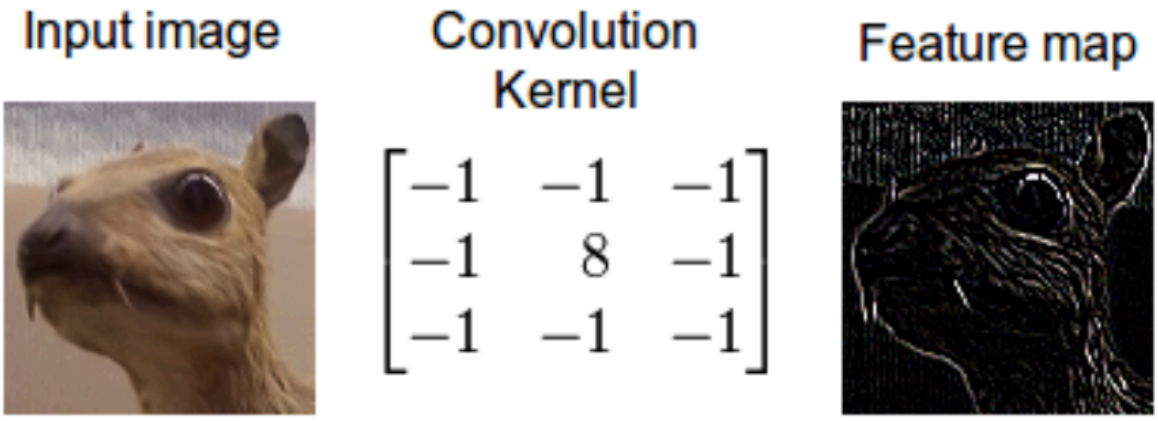

Figure 8 : Convolving an image with an edge detector kernel

Kernel:

The kernel, alternatively referred to as the filter, comprises a compact array of numerical values that traverses the input (whether it's an image or signal) during the convolution operation. The primary objective is to implement localized operations on the input, enabling the detection of specific features. Within the kernel, each element signifies a weight that undergoes multiplication with the corresponding input value during convolution. Enclosed in Figure 9 below are various illustrative examples.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 9 : shows us various convolution images after applying different types of filters or Kernels.

Stride:

Stride refers to the displacement magnitude of the kernel as it traverses the input during convolution. When the stride is set to 1, the kernel moves one position at a time, whereas a stride of 2 causes the kernel to skip two positions per movement. The chosen stride directly affects the spatial characteristics of the convolution output. Opting for larger strides can diminish the dimensionality of the output, whereas smaller strides preserve more spatial details. Larger strides contribute to computational efficiency, thereby accelerating the operation and potentially affecting its quality. In Figure 10 depicted below, a kernel marked in red navigates the pixel map of the image using a stride of 1.

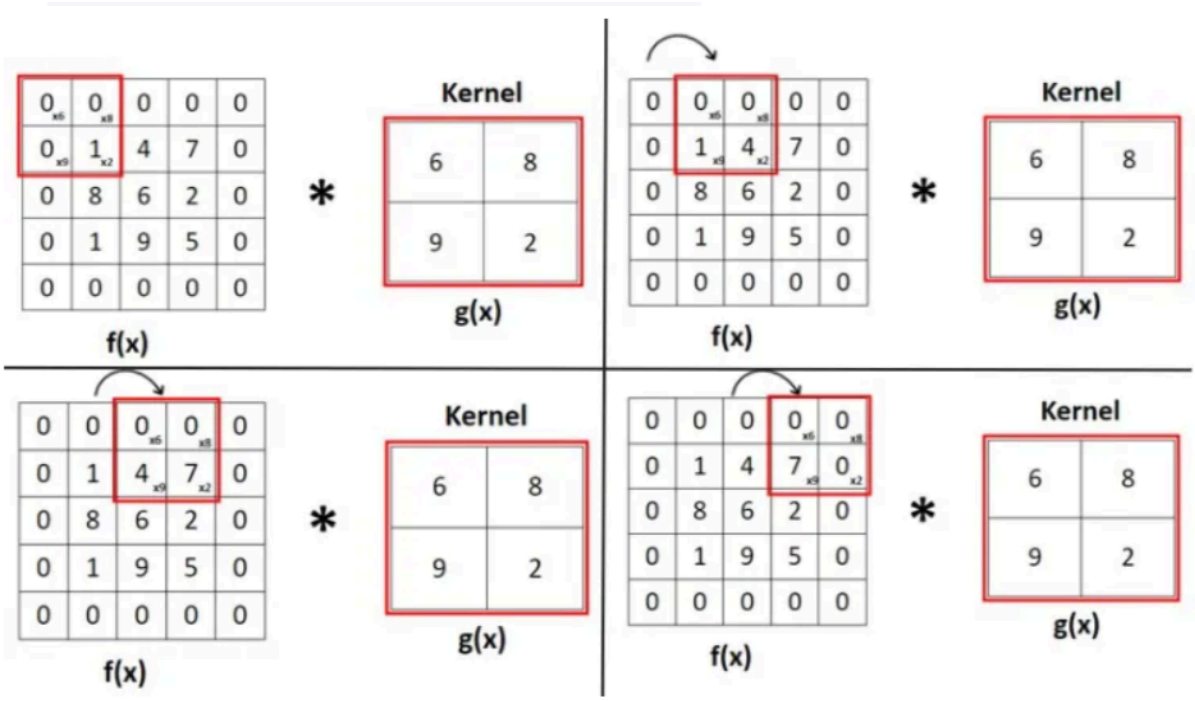

Figure 10: Example of Stride

Padding:

Padding, often referred to as "padding", involves the addition of extra pixels surrounding the edges of the input image, usually zeros, before conducting convolution operations. The purpose of this practice is to ensure uniform treatment of information across all regions of the image during convolution.

During the application of a filter (kernel) to an image, it typically traverses the image pixel by pixel. Without padding, pixels at the image edges have fewer neighboring pixels compared to those in the center, potentially resulting in information loss in these regions. As illustrated in Figure 11 below, padding serves to mitigate this issue by providing additional pixels around the image's perimeter.

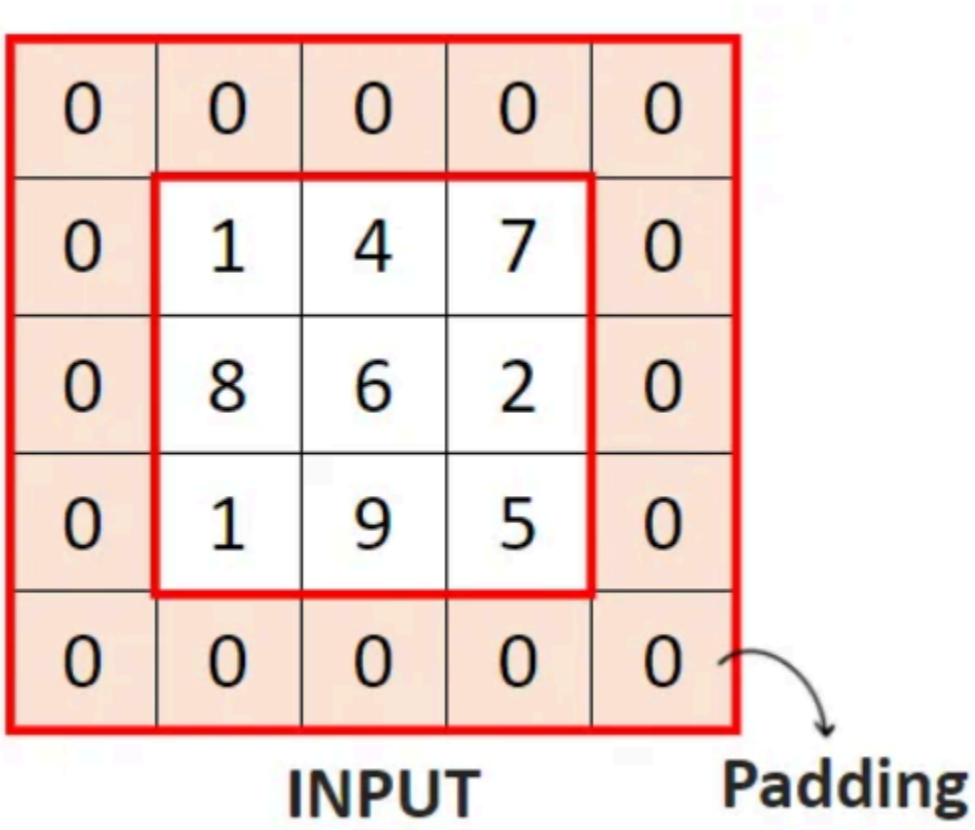

Figure 11: Example of zero padding

Specifically in the Yolov8 conv block:

2D Convolution:

During the 2D convolution operation, a filter is applied to the input to extract local features. Each position in the resulting feature map represents a weighted linear combination of values within the input's local region.

Batch Normalization (BatchNorm 2D):

Activations resulting from convolution are normalized through Batch Normalization. This process involves computing averages and standard deviations across the batch to stabilize the distribution of activations, aiding in faster convergence during training.

Application of the SiLU Function:

Following convolution and optionally Batch Normalization, the SiLU activation function is applied to the output. SiLU, defined as $SiLU(x) = x * \sigma(x)$, utilizes the logistic function (sigmoid) σ to introduce nonlinearity. This activation function enhances the model's capacity to learn nonlinear representations of data.

Propagation to Subsequent Layers:

The output of the SiLU function (or Batch Normalization, if applied subsequently) is then propagated to subsequent layers of the neural network. The introduction of nonlinearity, particularly through the SiLU function, plays a critical role in enabling the network to learn complex, nonlinear relationships within the data.

3.5 Workflow

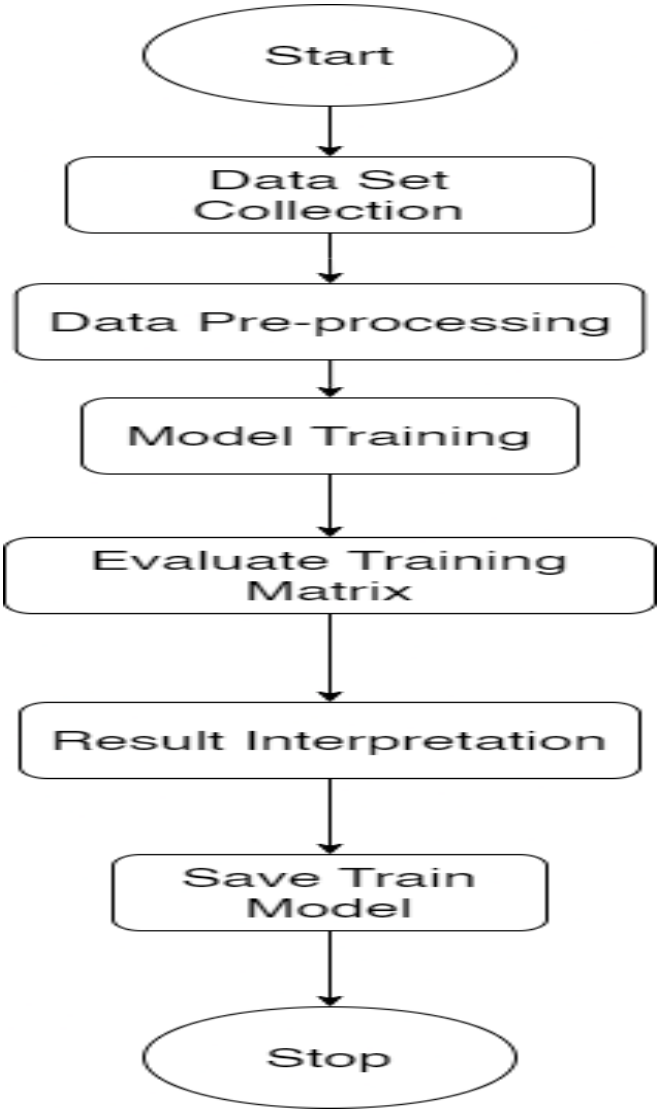

Figure 12: Work-flow Diagram

3.6 Evaluation protocol

We evaluate the model's efficiency by comparing the mean average precision (mAP) at a detection Intersection over Union (IOU) threshold of 0.5 (AP0.5). This evaluation protocol is widely used in several major detection benchmarks [22, 26, 27]. We conduct this evaluation specifically for key frames at a resolution of 2304 x 1296. To detect vehicles, we set the confidence threshold to 0.5, a standard metric for threshold metrics used in inference.

Additionally, we compare the inference speed of the models across various image resolutions. To achieve this, we scale the collected data frames to two other frame sizes: 720 x 576 and 3840 x 2160. The inference speeds at these down-sampled and up-sampled frame sizes are then compared with the original frame size of the dataset. This comparison provides insights into how the model's performance varies with different image resolutions

CHAPTER 4 : RESULTS

In preparing our dataset, we adhere to the principle of sample balance to address the challenge of disparate ratios between instances across categories or environmental conditions. Recognizing the abundance of vehicle instances during the day, we prioritize the collection of a substantial number of frames from night and rainy conditions to ensure a balanced representation of different weather scenarios.

However, our collected dataset exhibits a sub-sampled characteristic of today's real-world traffic, with a notably higher number of instances for cars and motorcycles. Conversely, instances of trucks and buses are comparatively fewer, likely due to their larger presence in an image frame. Despite their lower frequency, the instances of trucks and buses should be sufficient to capture relevant features for these categories in future model training and research endeavors.
A summary of the overall dataset statistics is provided in Table 1.

Table 1: Distribution of data in Nepali Traffic Dataset (NTD)

	Rainy	Peak Time	Night	Total
Total Instances	8332	5319	7553	21204
Car	3459	1849	3337	8645
Bus	974	471	568	2013
Truck	116	119	109	344
Motorcycle	3664	2780	3503	9947
Bicycle	119	100	36	225

The inference time of the YOLOv3 model exhibits a slight improvement on lower resolution test frames. This observation suggests that the YOLOv3 model can be effectively utilized without the need to alter the frame resolution. Importantly, this indicates that the model's performance remains relatively stable across varying resolutions, without experiencing haphazard degradation. Consequently, users can deploy the YOLOv3 model confidently across different resolutions without compromising performance.

Table 2: Results for 10 epoch

mAP50	Rainy	PeakTime	Night
Car	0.175	0.18	0.185
Bus	0.104	0.095	0.107
Truck	0.0143	0.012	0.015
Motorcycle	0.0971	0.1	0.09
bicycle	0	0	0
overall	0.0781	0.07	0.0849

The table shows the mean Average Precision at 50% (mAP50) for different types of vehicles under various conditions: Rainy, Peak Time, and Night. The mAP50 is a metric that measures how well a model can detect objects in images. A higher mAP50 means a better performance.
Car detection: The model performs the best on detecting cars under all conditions, with the highest mAP50 of 0.185 at night. This suggests that the model is good at recognizing the shape and features of cars, even in low-light situations.
Bus and Truck detection: The model performs poorly on detecting buses and trucks, with the lowest mAP50 of 0.012 for trucks in peak time. This suggests that the model struggles to distinguish between these large vehicles, especially when they are crowded or occluded by other objects.
Motorcycle detection: The model performs moderately on detecting motorcycles, with a slightly higher mAP50 of 0.1 in peak time than in rainy or night conditions. This suggests that the model can detect motorcycles better when they are more visible or contrasted with the background.
Bicycle detection: The table does not provide any data for bicycle detection, which means that the model either did not detect any bicycles in the images or the results were too low to be reported. This suggests that the model is unable to recognize bicycles, possibly because they are too small or similar to other objects.

Table 3: Results for 100 epoch

mAP50	Rainy	PeakTime	Night
Car	0.715	0.749	0.705
Bus	0.756	0.648	0.758
Truck	0.815	0.493	0.581
Motorcycle	0.525	0.692	0.51
bicycle	0.288	0.142	0.106
overall	0.62	0.545	0.532

The table uses a metric called mAP50, which stands for mean Average Precision at 50%. This metric measures how well the model can locate and identify the vehicles in the images. A higher mAP50 means a more accurate and reliable model.

Improvement from 10 epoch: The table shows that the model improved significantly from 10 epoch to 100 epoch in terms of mAP50 scores. For example, the mAP50 for car detection in rainy condition increased from 0.175 to 0.715, which is more than four times better. This suggests that the model benefited from more training and learned to recognize the vehicles better.

Best and worst conditions: The table shows that the model performed the best in rainy condition, with an overall mAP50 of 0.62. This could be because the raindrops create more contrast and edges in the images, which help the model detect the vehicles. The model performed the worst in night condition, with an overall mAP50 of 0.532. This could be because the darkness reduces the visibility and quality of the images, which make the model less confident.

Best and worst vehicle types: The table shows that the model performed the best on detecting trucks, with the highest mAP50 of 0.815 in rainy condition. This could be because trucks are large and distinctive, which make them easier to spot. The model performed the worst on detecting bicycles, with the lowest mAP50 of 0.106 in night condition. This could be because bicycles are small and similar to other objects, which make them harder to distinguish.

Table 4 : Results with and without background image

mAP50	Without Background	With Background
Car	0.175	0.245
Bus	0.104	0.139
Truck	0.0143	0.0294
Motorcycle	0.0971	0.131
bicycle	0	0.000271
overall	0.0781	0.109

The table uses a metric called mAP50, which measures how well the model can locate and identify the vehicles. A higher mAP50 means a more accurate and reliable model.

The table has three columns,mAP50, Without Background, and With Background. The mAP50 column lists the types of vehicles that the model can detect: Car, Bus, Truck, Motorcycle, and Bicycle. The Without Background and With Background columns show the mAP50 values for each type of vehicle in those conditions. The table also has a row for the overall average mAP50 value for both conditions.

The table shows that the model performs better when there is a background in the images than when there is not. The mAP50 values are higher for all types of vehicles except bicycles when there is a background. The table also shows that the model performs the best on detecting trucks and the worst on detecting bicycles. The overall average mAP50 value is 0.109 with background and 0.0781 without background.

Table 5 : Results with all data set combined

objects	Precision	Recall	map50
Car	0.902	0.86	0.916
Bus	0.862	0.862	0.907
Truck	0.893	0.911	0.961
Motorcycle	0.82	0.791	0.838
bicycle	0.83	0.39	0.516
overall	0.861	0.763	0.828

This image is a table that shows the performance of a machine-learning model that can detect different types of vehicles in images. The table uses three metrics to evaluate the model: precision, recall, and map50. Precision measures how many of the detected vehicles are correct, recall measures how many of the actual vehicles are detected, and map50 measures how well the model can locate and identify the vehicles. A higher value for each metric means a better performance.

Overall performance: The table shows that the model has an overall precision of 0.861, an overall recall of 0.763, and an overall map50 of 0.828. These are relatively high scores, indicating that the model is effective and reliable in detecting vehicles in images.

Best and worst vehicle types: The table shows that the model performs the best on detecting trucks, with the highest precision of 0.893, the highest recall of 0.911, and the highest map50 of 0.961. This suggests that the model is good at recognizing the shape and features of trucks, even when they are partially occluded or blended with the background.

The table shows that the model performs the worst on detecting bicycles, with the lowest precision of 0.83, the lowest recall of 0.39, and the lowest map50 of 0.516. This suggests that the model struggles to distinguish between bicycles and other objects, especially when they are small or similar to the background.

Trade-off between precision and recall: The table shows that there is a trade-off between precision and recall for some vehicle types. For example, the model has a high precision but a low recall for motorcycles, meaning that it detects few motorcycles but most of them are correct. On the other hand, the model has a low precision but a high recall for buses, meaning that it

detects many buses but some of them are wrong. This trade-off reflects the difficulty of balancing between sensitivity and specificity in object detection.

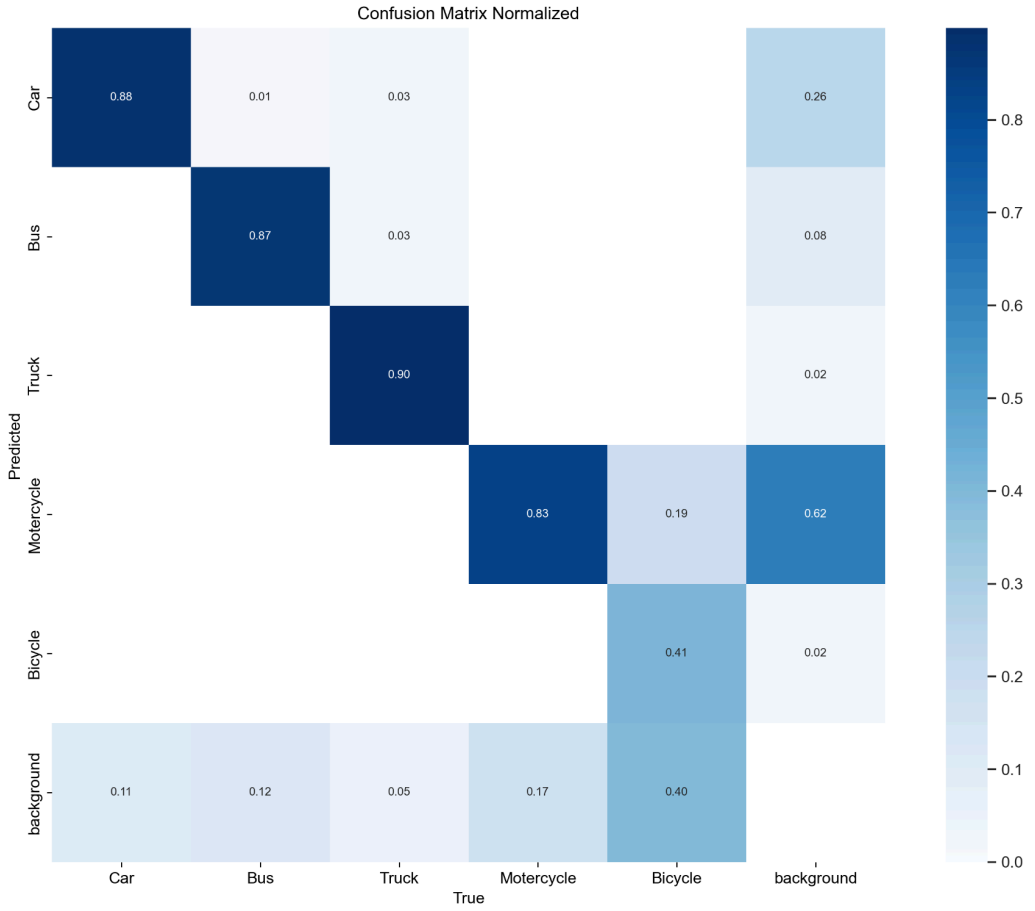

Figure 13: Confusion matrix normalized

The presented visual representation is a tabular assessment of a machine learning model's performance in detecting various vehicle types and background elements. The evaluation employs three key metrics: precision, recall, and mAP50. Precision gauges the accuracy of identified vehicles, recall measures the model's ability to capture actual vehicles, and mAP50 assesses the model's effectiveness in localizing and identifying vehicles. Higher metric values indicate superior model performance.

The table encompasses six distinct categories: Car, Bus, Truck, Motorcycle, Bicycle, and Background. These categories are delineated on both the x-axis (representing predicted labels) and y-axis (representing true labels). The color gradient within the table employs darker shades

of blue to signify higher metric values and lighter shades for lower values. Diagonal cells from the top left to the bottom right denote true positive rates for each category, reflecting instances where the model's predictions align with actual labels. Conversely, off-diagonal cells represent model errors, such as instances where the model predicted "Car" but the true label was "Bus." A color scale to the right of the matrix provides a visual reference for values ranging from 0.0 to 0.8.

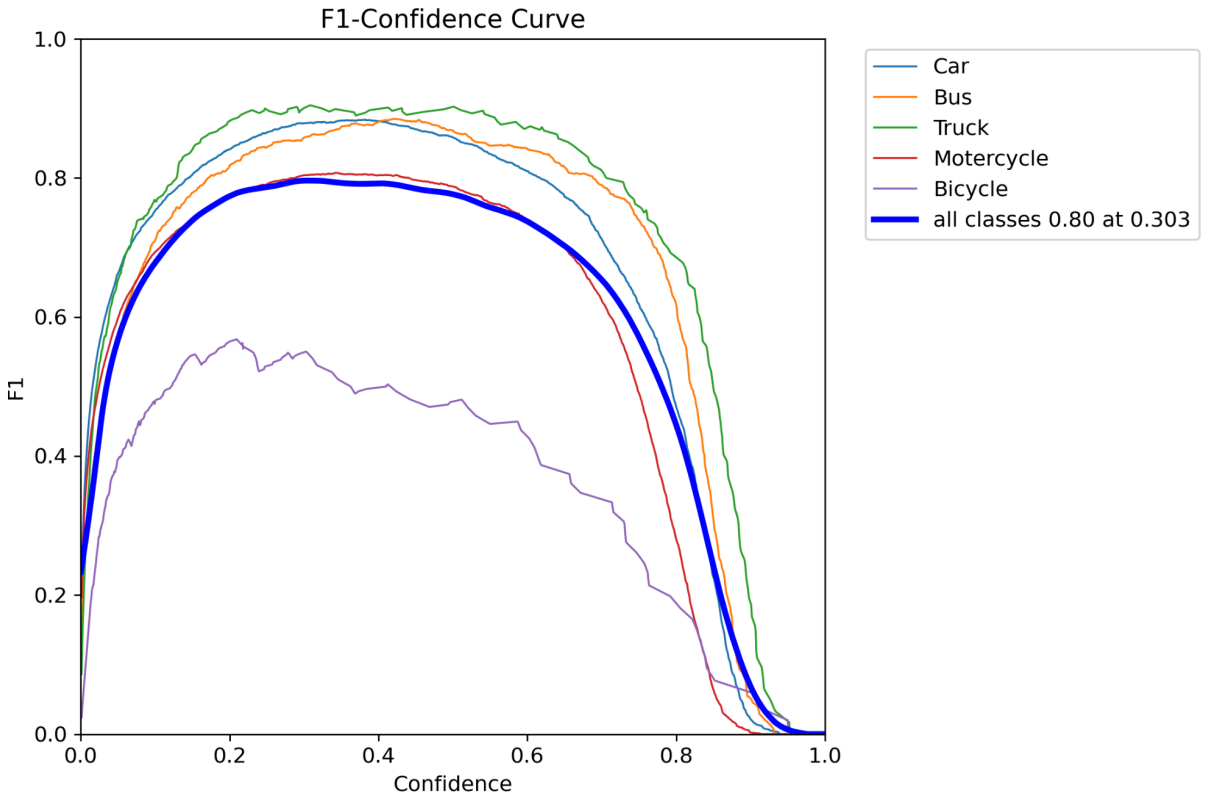

Figure 14: F1-Confidence Curve

The graph portrays the F1 score of a machine learning model designed for the detection of various vehicle types and background elements within images. The x-axis delineates the confidence level associated with the model's predictions, ranging from 0 to 1, while the y-axis represents the F1 score, an indicator of the model's accuracy that integrates precision and recall.

Confidence level: Denotes the degree of certainty exhibited by the model in its predictions, with higher values indicating increased confidence.

F1 score: Reflects the accuracy of the model in identifying vehicles and background elements within the images. A higher F1 score signifies greater accuracy, achieved through a harmonic mean of precision and recall metrics.

Optimal point: Represents the juncture where the F1 score attains its peak for each vehicle type or background category. This point signifies an optimal compromise between precision and recall, influenced by the complexity and frequency of detection for each category.

All classes: Noted by black text, all categories achieve an F1 score of 0.80 at a confidence level of approximately 0.303. This suggests a commendable performance, indicating an 80% accuracy in detecting various objects when the model exhibits a confidence level of 30.3%. However, this also implies room for enhancement by augmenting the model's confidence and accuracy.

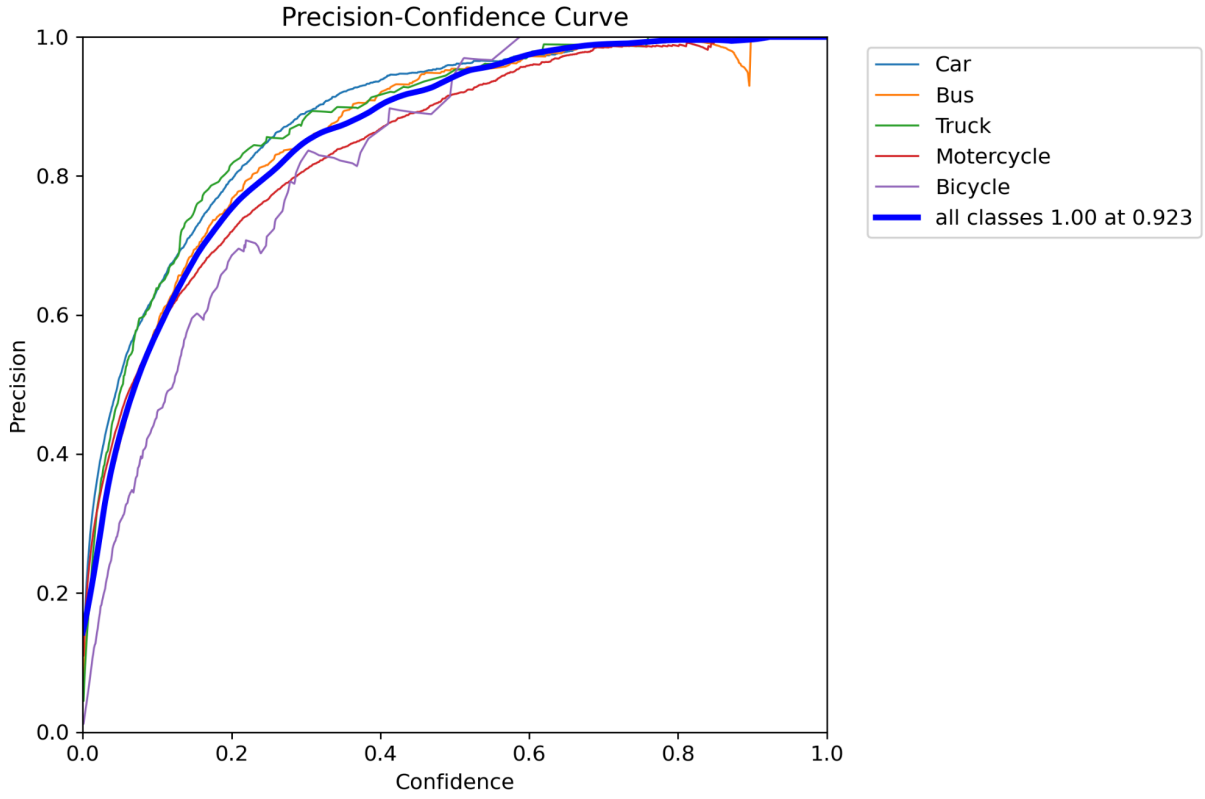

Figure 15 : Precision-Confidence Curve

The graph presents the precision of a machine learning model dedicated to detecting various vehicle types and background elements within images. The x-axis illustrates the confidence level associated with the model's predictions, ranging from 0 to 1, while the y-axis represents precision, a metric indicating the accuracy of the model in correctly identifying objects.

Confidence level: Denotes the level of certainty exhibited by the model in its predictions, with higher values signifying greater confidence.
Precision: Reflects the accuracy of the model in object detection, measured by the ratio of true positives to the sum of true positives and false positives.
Optimal point: Represents the juncture where precision reaches its peak for each vehicle type or background category. This point indicates the most favorable balance between confidence and accuracy for the model, varying based on the complexity and frequency of detection for each category.
All classes: Noted by black text, all categories achieve a precision of 1.00 at a confidence level of 0.923. This denotes that the model can detect all vehicle types and background elements with 100% accuracy when it attains a confidence level of 92.3%. Such performance underscores the model's exceptional effectiveness and reliability.

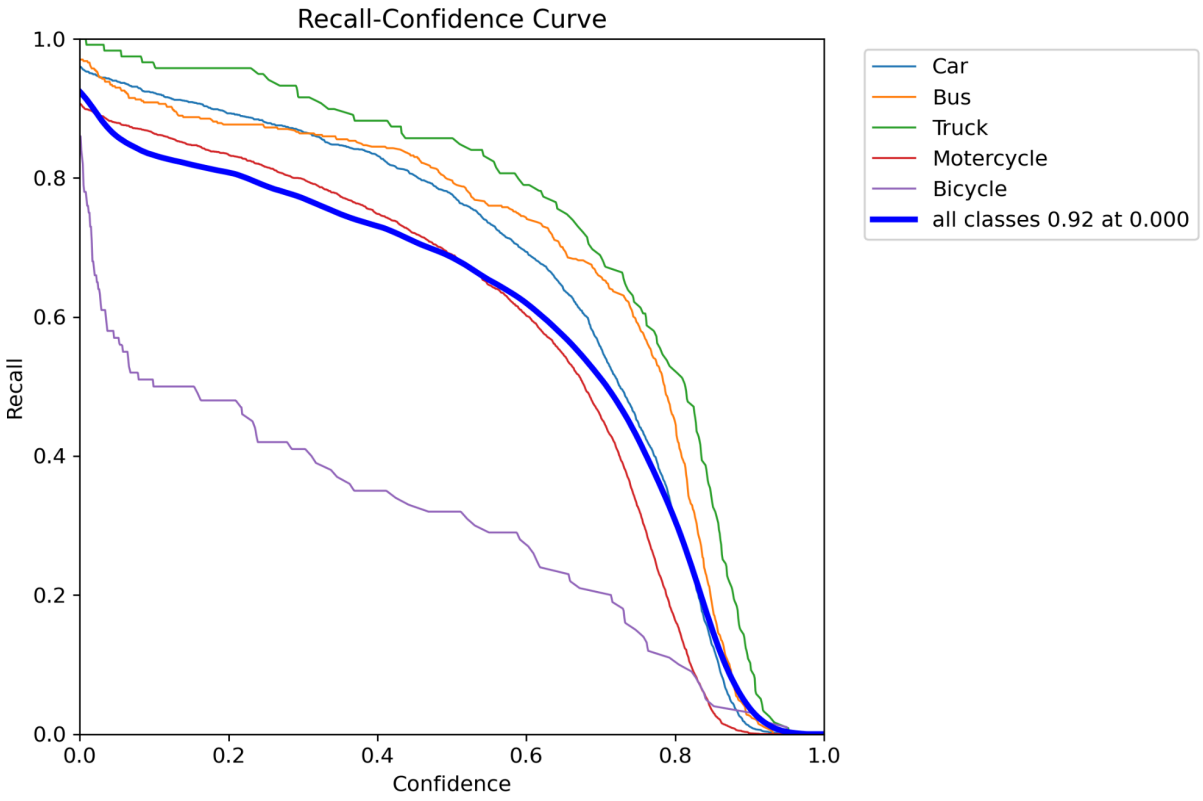

Figure 16: Recall-Confidence Curve

The graph depicts the recall of a machine learning model tasked with detecting various vehicle types and background elements within images. The x-axis denotes the confidence level associated with the model's predictions, ranging from 0 to 1, while the y-axis represents recall, which measures the model's sensitivity in identifying objects.

Confidence level: Indicates the level of certainty exhibited by the model in its predictions, with higher values suggesting greater confidence.
Recall: Reflects the sensitivity of the model in object detection, calculated by the ratio of true positives to the sum of true positives and false negatives.
Optimal point: Represents the juncture where recall achieves its maximum for each vehicle type or background category. This point signifies the most favorable balance between confidence and sensitivity for the model, influenced by the complexity and frequency of detection for each category.
All classes: Dark blue text highlights that all categories achieved a recall of 0.92 at a confidence level of 0.000. This indicates that the model can detect 92% of all vehicle types and background elements in images even when it exhibits no confidence in its predictions. This surprising result suggests that the model is highly sensitive but lacks specificity.

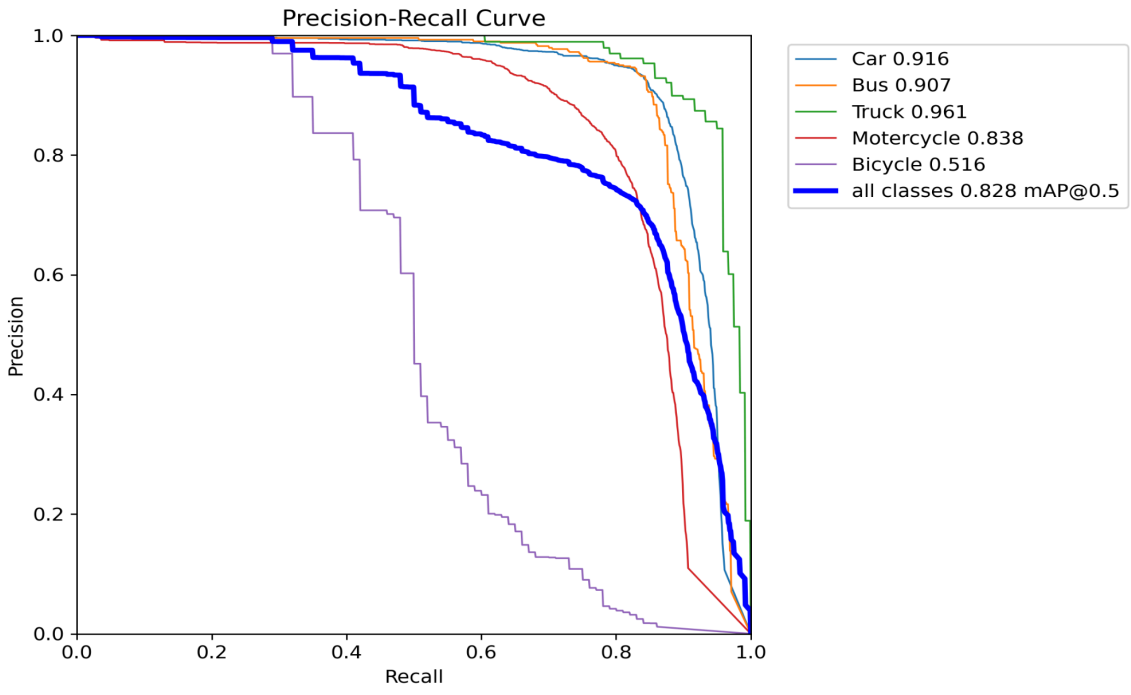

Figure 17: Precision-Recall Curve

This graph depicts the performance evaluation of a machine learning model in identifying various types of vehicles in images, measured by precision and recall metrics. Precision represents the proportion of correct predictions among all predictions made, while recall signifies the proportion of correct predictions among all relevant instances present. The graph also illustrates the average precision score for each vehicle category, condensing the precision-recall curve into a single metric. Such visualizations serve as crucial tools for assessing the model's efficacy and comparing it against alternative methodologies. Observations gleaned from the graph include the model's exemplary performance in discerning buses, evidenced by its highest precision and recall rates, along with an outstanding average precision score of 0.967. Conversely, the model struggles notably in accurately identifying bicycles, with the lowest precision and recall rates, and an average precision score of 0.516. Furthermore, an analysis of the model's performance regarding trucks reveals a notable trend: while it exhibits commendable precision, it concurrently suffers from subpar recall. This indicates the model's proficiency in correctly identifying trucks among its predictions, albeit at the expense of missing several actual instances present in the images. In contrast, the model demonstrates a nuanced performance for motorcycles, characterized by relatively low precision but high recall. While adept at detecting most motorcycles present, it concurrently generates numerous false positive predictions for unrelated objects, thus impeding its precision score. For the car category, the model showcases moderate performance, marked by a balanced interplay between precision and recall, and an average precision score of 0.916. Overall, the model's aggregate performance, quantified by the mean average precision (mAP) score at a threshold of 0.5, stands at 0.828. This signifies the model's ability to accurately identify approximately 83% of vehicles present in the images on average.

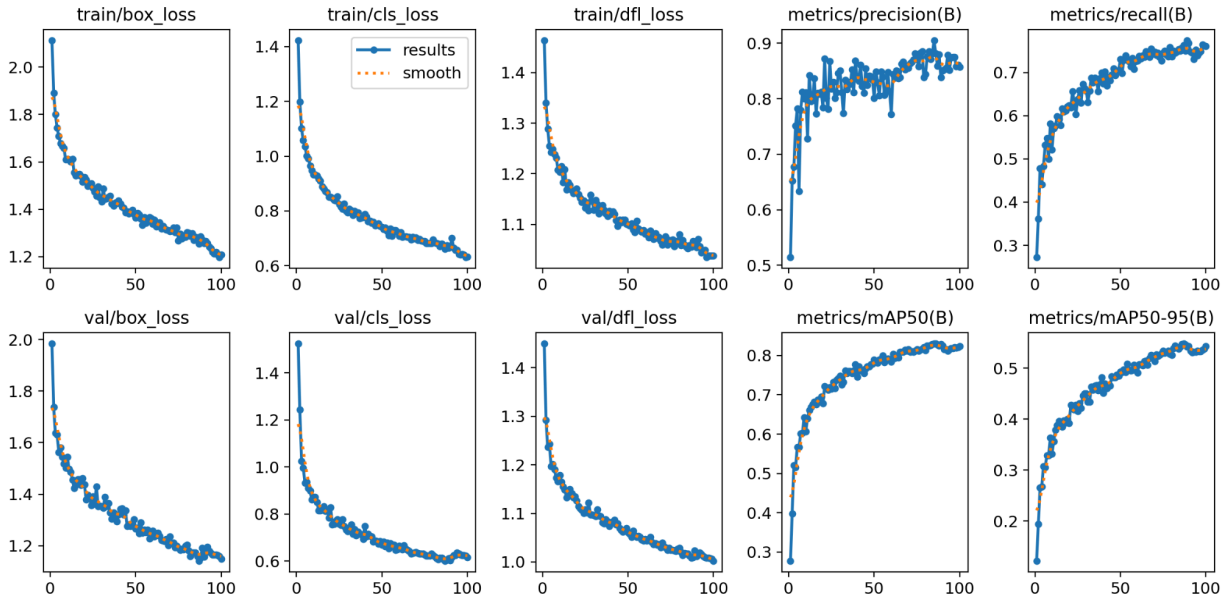

Figure 18: Results over 100 epoch

The graph depicts the recall of a machine learning model tasked with detecting various vehicle types and background elements within images. The x-axis denotes the confidence level associated with the model's predictions, ranging from 0 to 1, while the y-axis represents recall, which measures the model's sensitivity in identifying objects.

Confidence level: Indicates the level of certainty exhibited by the model in its predictions, with higher values suggesting greater confidence.
Recall: Reflects the sensitivity of the model in object detection, calculated by the ratio of true positives to the sum of true positives and false negatives.
Optimal point: Represents the juncture where recall achieves its maximum for each vehicle type or background category. This point signifies the most favorable balance between confidence and sensitivity for the model, influenced by the complexity and frequency of detection for each category.
All classes: Dark blue text highlights that all categories achieved a recall of 0.92 at a confidence level of 0.000. This indicates that the model can detect 92% of all vehicle types and background elements in images even when it exhibits no confidence in its predictions. This surprising result suggests that the model is highly sensitive but lacks specificity.

Figure 18 : Model output on test image

CHAPTER 5 : CONCLUSION AND FURTHER WORK

5.1 Conclusion

In this study, we introduce a novel custom dataset focused on Nepali traffic, aptly named the Nepali Traffic Dataset (NTD). Our dataset is designed to offer a valuable resource for research and experimentation in the field of traffic analysis and related areas. The NTD holds potential for utilization in diverse experiments, serving as an alternative database or a challenging research collection.

The table 3 shows that the model performed the best in rainy conditions, with an overall mAP50 of 0.62. This could be because the raindrops create more contrast and edges in the images, which help the model detect the vehicles. The model performed the worst in night conditions, with an overall mAP50 of 0.532. This could be because the darkness reduces the visibility and quality of the images, which make the model less confident

The table 4 shows that the model performs better when there is a background in the images than when there is not. The mAP50 values are higher for all types of vehicles except bicycles when there is a background. The table also shows that the model performs the best on detecting trucks and the worst on detecting bicycles. The overall average mAP50 value is 0.109 with background and 0.0781 without background.

The table 5 shows that the model has an overall precision of 0.861, an overall recall of 0.763, and an overall map50 of 0.828. These are relatively high scores, indicating that the model is effective and reliable in detecting vehicles in images.

5.2 Further Works

As part of our future work, we plan to leverage a foggy dataset for training purposes, in conjunction with our custom Nepali Traffic Dataset (NTD). By incorporating foggy weather conditions into our training data and conducting hyperparameter tuning, we aim to develop a novel model capable of detecting traffic vehicles in Nepal with enhanced accuracy.

Through the integration of diverse weather conditions into our training regimen, we anticipate creating a more robust model that can effectively handle varying environmental challenges commonly encountered in Nepali traffic scenarios. Our ultimate goal is to craft a highly accurate and adaptable model that can be deployed across different areas of traffic management systems,

aiding in tasks such as vehicle detection, monitoring, and control. This endeavor aligns with our commitment to advancing the field of traffic analysis and contributing to the development of practical solutions tailored to the specific context of Nepal

CHAPTER 6 : REFERENCES

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[2]Bochkovsky, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.

[3]Redmon, J. (2013). Darknet: Open Source Neural Networks in C. Retrieved from https://pjreddie.com/darknet/

[4] Abadi, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16).

[5] Li, Y., et al. (2018). A Vehicle Detection Algorithm Based on Deep Learning in UAV Images. IEEE Access, 6, 55774-55785.

[6] Chen, X., et al. (2019). Vehicle Detection in Aerial Images Based on Region Convolutional Neural Networks. Remote Sensing, 11(12), 1444.

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[8] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems.
[9] M. Tahailyas, "Principles of YOLO V8," Medium, 2022.

[10] S. Z. Ali, "Principles of YOLOv8," Medium, 2022

[11] "Ultralytics." Ultralytics Documentation, Ultralytics, Available:

https://docs.ultralytics.com/pt/

[12] Ultralytics, "YOLOv5 Architecture Description," Ultralytics Documentation, Year. [Online].

Available: https://docs.ultralytics.com/yolov5/tutorials/architecture_description/

[13] "YOLOv8 Architecture Detailed Explanation - A Complete Breakdown" YouTube, Available: https://www.youtube.com/watch?app=desktop&v=HQXhDO7COj8.

[14] NVIDIA, "Convolution," NVIDIA Developer. Available: https://developer.nvidia.com/discover/convolution.

[15] "What's New in YOLOv8?," Roboflow Blog Available: https://blog.roboflow.com/whats-new-in-yolov8/.

[16] Ahmad Arib Alfarisy, Quan Chen, and Minyi Guo. Deep learning based classification for paddy pests & diseases recognition. In Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence, pages 21–25, 2018.

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look onceUnified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

[19] Ross Girshick. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, andJian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497, 2015.

[21] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. arXiv preprint arXiv:1605.06409, 2016.

[22] K He, G Gkioxari, P Doll´ar, and RB Girshick. Mask r-cnn. corr, abs/1703.06870. arXiv preprintarXiv:1703.06870, 2017.

[23] Tingyu Mao, Wei Zhang, Haoyu He, Yanjun Lin, Vinay Kale, Alexander Stein, and Zoran Kostic, Aic2018 report: Traffic surveillance research. In Proceedings of the IEEE Conference Computer Vision and Pattern Recognition Workshops, pages 85,92, 2018.

[24] Aleksandr Fedorov, Kseniia Nikolskaia, Sergey Ivanov, Vladimir Shepelev, and Alexey Minbaleev.Traffic flow estimation with data from a video surveillance camera. Journal of Big Data, 6(1):73,2019.

[25] Yaoming Zhang, Xiaoli Song, Mengen Wang, Tian Guan, Jiawei Liu, Zhaojian Wang, Yajing Zhen, Dongsheng Zhang, and Xiaoyi Gu. Research on visual vehicle detection and tracking based on deep learning. In IOP Conference Series: Materials Science and Engineering, volume 892, page 012051.IOP Publishing, 2020.

[26] Fei Liu, Zhiyuan Zeng, and Rong Jiang. A video based real-time adaptive vehicle-counting system for urban roads. PloS one, 12(11):e0186098, 2017.

[27] Guizhen Yu, Sifen Wang, Mingxing Li, Yaxin Guo, and Zhangyu Wang. Vision-based vehicle detection in foggy days by convolutional neural network. In Chinese Intelligent Systems Conference, pages 334 343. Springer, 2019.

[28] Md Nasim Khan and Mohamed M Ahmed. Trajectory-level fog detection based on in-vehicle video camera with tensorflow deep learning utilizing shrp2 naturalistic driving data. Accident Analysis & Prevention, 142:105521, 2020.

[29] Xuewen Chen, Huaqing Chen, and Huan Xu. Vehicle detection based on multifeature extraction and recognition adopting rbf neural network on adas system. Complexity, 2020, 2020.

[30] Tuan-Anh Pham and Myungsik Yoo. Nighttime vehicle detection and tracking with occlusion handling by pairing headlights and taillights. Applied Sciences, 10(11), 2020.