KATHFORD INTERNATIONAL COLLEGE OF ENGINEERING AND MANAGEMENT

Balkumari, Lalitpur

A

Major Project Report

On

"VEHICLE SPEED ESTIMATION USING DEEP LEARNING"

[Subject Code: EX 755]

Project Members

Anish Thapa Magar (KIC076BEI002)

Neha Adhikari (KIC076BEI012)

Subhadra Osthi (KIC076BEI015)

DEPARTMENT OF COMPUTER AND ELECTRONICS & COMMUNICATION ENGINEERING

LALITPUR, NEPAL

MARCH, 2024

KATHFORD INTERNATIONAL COLLEGE
OF ENGINEERING AND MANAGEMENT
(Affiliated to Tribhuvan University)

"VEHICLE SPEED ESTIMATION USING DEEP LEARNING"

A PROJECT REPORT
SUBMITTED TO THE DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN ELECTRONICS, COMMUNICATION, AND INFORMATION ENGINEERING

Submitted By:

Anish Thapa Magar (KIC076BEI002)

Neha Adhikari (KIC076BEI012)

Subhadra Osthi (KIC076BEI015)

Submitted To:

DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

MAY, 2023

COPYRIGHT

LETTER OF APPROVAL

The undersigned certify that they have read and recommended to the Department of Computer and Electronics Engineering for acceptance, a project entitled "**VEHICLE SPEED ESTIMATION USING DEEP LEARNING**", submitted by **Anish Thapa Magar**, **Neha Adhikari**, and **Subhadra Osthi** in partial fulfillment of the requirement for the award of the degree of "**Bachelor in Engineering of Electronics, Communication and Engineering**".

External Examiner

…….

Prof. Dr. Ram Krishna Maharjan

Professor

Department of Electronics & Communication Engineering

Institute of Engineering, Pulchowk, Tribhuvan University

Project Supervisor

…….

Er. Saban Kumar KC

Sr. Lecturer

Department of Computer and Electronics

Kathford International College of Engineering and Management

DATE OF APPROVAL: 7.03.2024

KATHFORD
INTERNATIONAL COLLEGE OF ENGINEERING & MANAGEMENT || AFFILIATED TO TRIBHUVAN UNIVERSITY

Balkumari, Lalitpur
01-5201241, 01-5201911
(Engineering Block)
01-5186046

DEPARTMENTAL ACCEPTANCE

The project entitled "**VEHICLE SPEED ESTIMATION USING DEEP LEARNING**", submitted by Anish Thapa Magar Neha Adhikari, and Subhadra Osthi a project report submitted to the department of computer and electronics engineering in partial fulfillment of the requirements for the degree of "Bachelor in Electronics, Communication, and Information Engineering" has been accepted as a bona fide record of work carried out by them in department.

…….

Er. Chaitya Shova Shakya

Head of the Department

Department of Computer and Electronics & Communication Engineering Kathford International College of Engineering and Management

Balkumari, Lalitpur,

Nepal

ACKNOWLEDGEMENT

We use this opportunity to express our sincere gratitude to everyone who supported us throughout the course of this project. We are thankful for their aspiring guidance, invaluably constructive criticism, and friendly advice during the project work. We are grateful to them for sharing their views on issues related to the project.

We would like to thank the **Department of Computer and Electronics and Communication**, Kathford International College of Engineering and Management, Balkumari, Lalitpur for giving us this golden opportunity and encouraging us to work on this project. We are very thankful to our project coordinator & supervisor, **Er. Saban Kumar KC** for his continuous support, guidance, and supervision of this project. We express our gratitude to the Head of the Department of Computer and Electronics and Communication **Er. Chaitya Shobha Shakya** for her regular support and understanding.

We would also like to thank each person who mentored us directly or indirectly during the course of the project. Thanks, are also due to all our colleagues and everyone who has helped us out for the betterment of the project.

ABSTRACT

Accurate vehicle speed estimation is crucial for traffic monitoring and management. This research introduces a novel two-step approach that leverages YOLO for object detection and Recurrent Neural Networks (RNN), specifically Long Short-Term Memory (LSTM) networks, for estimating vehicle speed. The methodology was validated using the VS13 dataset, which contains 400 videos of single vehicles, with 80 reserved for testing. Initially, YOLO detects the vehicle and provides bounding boxes, from which the change in area is calculated across frames. This data is then processed by the LSTM network to estimate the vehicle's speed. During training, the model's RMSE significantly decreased from 60 to approximately 4 over 600 epochs. The final RMSE on the test set was 8.1010. When evaluated across 13 different vehicle models, the average RMSE was 5.7931, with individual vehicle RMSEs ranging from 4.0602 to 7.6198. Despite its accuracy, the model is limited to single-vehicle scenarios and requires 80 consecutive frames for effective speed estimation. Future work will focus on enhancing the model to handle multiple vehicles, improving user interface, and optimizing the algorithm for fewer frames. This research has particular relevance in the context of Nepal, where traffic monitoring can significantly benefit from cost-effective and efficient speed estimation systems.

Keywords: Vehicle Speed Estimation, YOLO, Deep learning, Artificial Intelligence (AI), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), RMSE,

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF ABBREVIATIONS

AI - Artificial Intelligence

CNN - Convolutional Neural Network

YOLO - You Only Look Once

MSE - Mean Squared Error

MAE - Mean Absolute Error

SGD - Stochastic Gradient Descent

OCR - Optical Character Recognition

AI - Artificial Intelligence

R-CNN - Region-based Convolutional Neural Networks

LSTM - Long Short-Term Memory

SSD - Single Shot-Multibox Detector

ROI - Regions of Interest

VGG - Visual Geometry Group

CHAPTER 1: INTRODUCTION

In the context of increasing urbanization and growing concerns about road safety and traffic management, the accurate estimation of vehicle speed plays a crucial role in ensuring efficient transportation systems. Nepal, like many other developing countries, faces challenges related to traffic congestion, road accidents, and inadequate infrastructure. In such contexts, the deployment of advanced technologies for speed estimation and traffic monitoring becomes imperative to address these challenges effectively.

1.1 Background

Nepal, nestled in the heart of the Himalayas, is characterized by diverse terrain, ranging from rugged mountainous regions to bustling urban centers. Despite its geographical challenges, Nepal experiences significant vehicular traffic, particularly in urban areas. However, the lack of robust traffic management systems and the absence of accurate speed estimation methods pose challenges in ensuring road safety and optimizing traffic flow.

Traditional methods of speed measurement, such as radar guns and manual speed traps, are often resource-intensive, time-consuming, and limited in their coverage. Moreover, these methods may not provide real-time data necessary for effective traffic management and decision-making. Consequently, there is a growing need for automated systems capable of accurately estimating vehicle speed from video data in real time.

1.2 Problem Statement

The existing challenges in traffic management and road safety in Nepal underscore the need for innovative solutions for speed estimation and traffic monitoring. Conventional methods are often inadequate in addressing the complex dynamics of traffic flow and the diverse terrain of the country. Additionally, the lack of scalable and efficient technologies hinders efforts to improve road safety and reduce traffic congestion

1.3 Objectives

1.3.1 Main Objective

Develop a robust and scalable model for vehicle speed estimation using deep learning techniques, specifically YOLO for object detection and LSTM networks for sequence learning, to address the challenges of traffic management and road safety in Nepal.

1.3.2 Specific Objective

1. Implement a deep learning architecture integrating YOLO for vehicle detection and LSTM for speed estimation.
2. Assess the model's accuracy and efficiency in the collected data.
3. Measure the model's effectiveness in improving traffic management and road safety, focusing on speed accuracy and reduction in road accidents.

1.4 Scope

1. The project includes the design and implementation of a deep learning architecture that integrates YOLO for accurate vehicle detection and tracking, followed by LSTM networks for precise speed estimation.
2. The project will exclusively address the task of estimating the speed of single vehicles present in the video frame.
3. Utilize existing public datasets only; no real-world parking lot data collection.
4. The project scope does not include the development or deployment of hardware systems for data collection or real-time monitoring.

CHAPTER 2: LITERATURE REVIEW

The field of vehicle speed estimation using video data has seen significant advancements over the past few years. Various methods have been explored, including traditional computer vision techniques and deep learning models. This literature review highlights key developments, comparing different approaches and their effectiveness in estimating vehicle speed.

2.1 Research and Related Work

Traditional Computer Vision Approaches

Traditional computer vision methods for vehicle speed estimation often rely on measuring the distance traveled by a vehicle over time or the time taken to travel a known distance. Keattisak Sangsuwan and Ekpanyapong (2020) demonstrated a method utilizing YOLOv3, DeepSORT, GoodFeatureToTrack, and the Pyramidal Lucas-Kanade optical flow algorithm to detect and track vehicles in video footage. By defining virtual intrusion lines on the road, the system measures pixel displacement and traveling time to estimate speed, achieving a mean absolute error (MAE) of 3.38 km/h and a root mean square error (RMSE) of 4.69 km/h.

David Fernández Llorca et al. (2021) provided a comprehensive survey on vision-based vehicle speed estimation techniques, emphasizing the cost benefits and potential for accurate identification without expensive range sensors. They categorized various methodologies and discussed the challenges and advantages of vision-based systems.

Deep Learning Approaches

Deep learning has revolutionized the field of vehicle speed estimation by providing more accurate and robust solutions. The YOLO (You Only Look Once) algorithm, introduced by Redmon et al. (2016), is a popular choice for object detection due to its real-time processing capabilities and high accuracy. YOLOv5, the latest iteration, continues to improve on these capabilities, making it a preferred tool for many researchers.

A notable study by Cvijetić, Djukanović, and Peruničić (2023) used YOLO for vehicle detection and a one-dimensional convolutional neural network (1D-CNN) for speed estimation. Their method involves calculating the change in the bounding box area

around the vehicle as it moves, achieving an average error of 2.76 km/h on the VS13 dataset.

Use of Recurrent Neural Networks

Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, are particularly effective for sequential data and time-series analysis. Hojjat Salehinejad et al. (2018) reviewed the advances in RNNs, highlighting their ability to learn long-term dependencies and their application in various fields including vehicle speed estimation. Similarly, Y. Bengio, P. Simard, and P. Frasconi (1994) discussed the challenges and benefits of using gradient descent for training RNNs, emphasizing the importance of capturing long-term dependencies in sequential data.

Datasets for Vehicle Speed Estimation

The availability of high-quality datasets is crucial for developing and validating vehicle speed estimation models. Djukanović, Bulatović, and Čavor (2022) introduced a dataset specifically designed for audio-video based vehicle speed estimation. This dataset includes recordings of 13 different vehicles at known speeds, providing a valuable benchmark for researchers.

The integration of YOLO for vehicle detection and LSTM for speed estimation presents a promising approach to vehicle speed estimation, particularly in the context of single-vehicle scenarios. While traditional methods offer foundational techniques, deep learning models provide enhanced accuracy and robustness. Future research should focus on addressing the limitations of current models, such as handling multiple vehicles in a frame and real-time implementation, to further advance the field.

2.2 Related Theory

2.2.1 CNN

Convolutional Neural Networks (CNNs) are a class of deep neural networks designed for processing structured grid data, such as images. They work by applying a series of filters to the image that extract features and recognize the pattern in the image. Then those extracted images are sent to a fully connected, dense layer to classify the image

according to the features in the image. CNNs also have a layer of non-linear activation function that allows the network to learn from complex data.

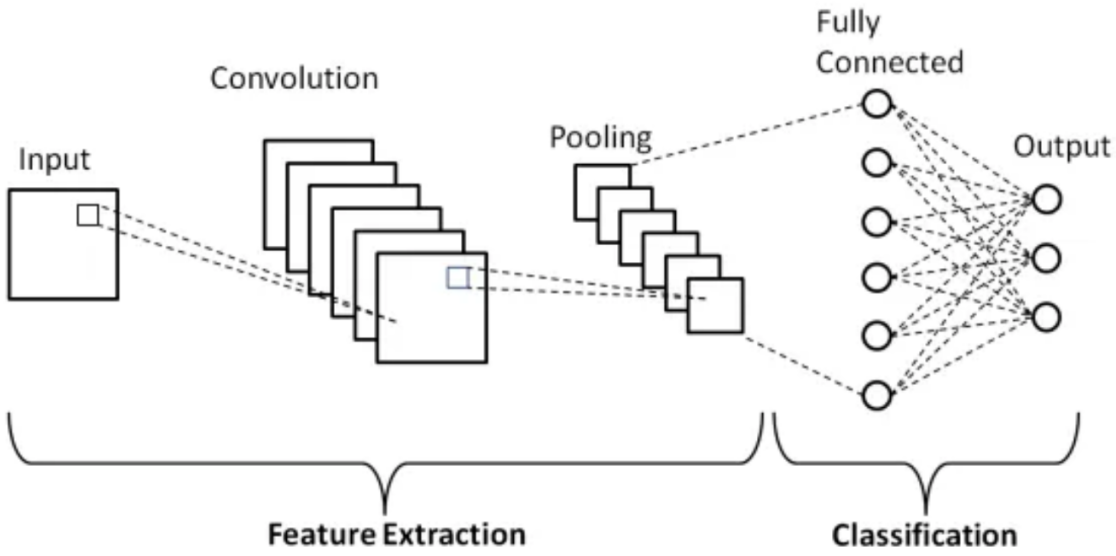


Figure 2. 1 CNN, Source: Binary Classification using CNN

In this paper, we used a CNN based architecture called YOLO.

2.2.2. Object Detection with YOLO

Object detection is a computer vision technique that involves identifying and locating objects within an image or a video. It combines classification and localization to detect the presence of multiple objects and determine their positions using bounding boxes. Object detection has numerous applications, including autonomous driving, surveillance, image retrieval, and augmented reality. Here we are using it measure speed of the vehicle.

There are two main groups of object recognition algorithms: one-stage and two-stage algorithms. The former ones detect and classify objects in a single forward pass, whereas the latter ones first preform object recognition and then, in the second pass, classify detected objects. Region based convolutional neural networks (R-CNNs) are classic examples of the two-stage algorithms

YOLO (You Only Look Once) is a real-time one-stage algorithm object detection system that has revolutionized the field with its speed and accuracy. Unlike traditional methods that apply a classifier to an image at multiple locations and scales, YOLO frames the detection problem as a single regression problem. This network divides the

image into regions and predicts bounding boxes and probabilities for each region simultaneously.

The YOLO model consists of multiple convolutional layers. It divides image in a grid, and for each grid cell it predicts multiple bounding boxes. The bounding box is a rectangular shape which encloses the detected object (see Fig. l). The bounding box predictions consist of five parameters: (x, y, h, w) and the box confidence score Parameters, x and y represent the coordinates of the box center, h and w the height and the width of the box, and the confidence score reflects how likely the box contains an object and how accurate is the predicted boundary box.

Since YOLO was introduced, many versions of it have been published, but the algorithm practically remains the same. In this paper, we use the 5th generation of YOLO, YOLOv5, known for its rapid detection, high accuracy, adaptability to different datasets, and a simple training procedure.

2.2.3 RNN

RNNs (Recurrent Neural Networks) are neural networks designed for sequential data processing. LSTMs (Long Short-Term Memory) are a type of RNN architecture designed to address the vanishing gradient problem, enabling better capture of long-term dependencies through memory cells and gating mechanisms.

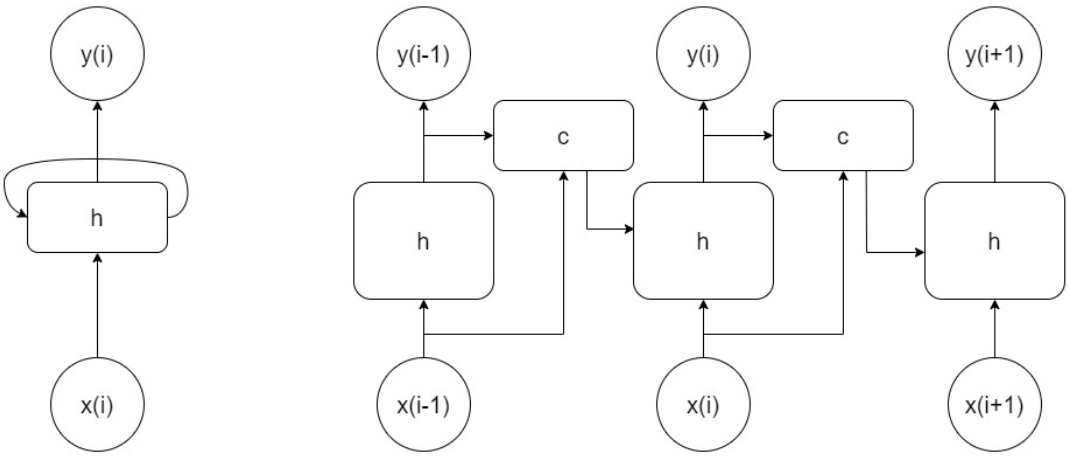


Figure 2. 2 Left: RNN Right: LSTM

2.2.3 Adam Optimizer

Adaptive Moment Estimation (ADAM) is a popular optimization technique used in neural networks Adam, an extension of SGD, optimizes neural networks by using adaptive learning rates and estimations of the first and second moments of the gradient. This approach prevents local minima trapping, controls update magnitudes, and incorporates bias correction terms to address initial stage biases, enhancing convergence stability and efficiency.

2.2.4 RMSE

Root Mean Square Error (RMSE) is a commonly used metric to measure the accuracy of a regression model. It represents the square root of the average squared differences between the predicted values and the actual values.

The formula for RMSE is:

$$\textbf{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

where,

- $\hat{y}_i$ is the predicted value for the i-th data point.
- y_i is the actual value for the i-th data point.
- n is the total number of data points.

2.3 Related Tools

2.3.1 Python

Python is a high-level, interpreted programming language known for its readability and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Widely used in web development, data science, automation, and artificial intelligence, Python has extensive libraries and a strong community. Some of the libraries using in this project are:

- Matplotlib, a comprehensive library for creating static, animated, and interactive visualizations in Python.

- OpenCV, a powerful library for computer vision, machine learning, and image processing tasks.
- Pandas, a versatile library for data manipulation and analysis, providing data structures like DataFrame and Series to efficiently handle structured data.
- Scikit-learn, a robust library for machine learning in Python, offering simple and efficient tools for data mining and data analysis.
- Tqdm is a fast, extensible library for adding progress bars to Python loops and tasks, enhancing user feedback during lengthy operations.

2.3.2 PyTorch

PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab. It offers dynamic computation graphs and intuitive APIs, making it popular for research and production. PyTorch supports GPU acceleration and integrates seamlessly with Python, facilitating easy model building and debugging.

2.4 VS13

The dataset used in this project is VS13 dataset by Đukanović, Slobodan & Bulatovic, Nikola & Čavor, Ivana (2022). It consists of on-road audio-video recordings of single vehicles passing a camera at known speeds, maintained stable by on-board cruise control. The dataset includes thirteen vehicles chosen to maximize diversity in manufacturer, production year, engine type, power, and transmission, resulting in 400 annotated audio-video recordings. Each video contains a single vehicle, implying that VS13 is intended for individual vehicle speed estimation, not the average traffic speed estimation. Recorded speeds in VS 13 range from 30 km/h to 110 km/h. During training and testing, 20% of the total videos i.e. 80 videos were held out for testing, whereas the remaining 320 are used for training and validating the model.

CHAPTER 3: METHODOLOGY

3.1 Development Model

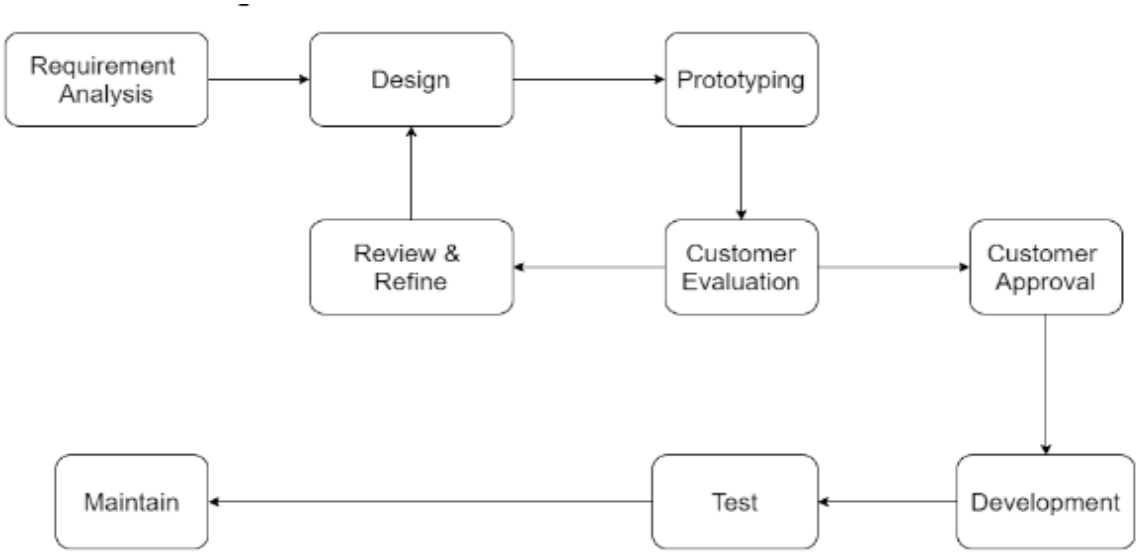


Figure 3. 1 Prototype Development Model

Prototyping is a software development methodology that focuses on the use of working models that are constantly refined based on feedback from the end user. Prototyping is most commonly used to develop systems with significant end-user interaction and complex user interfaces.

Because it allows for the capture of customer requirements at an early stage of development, software prototyping is becoming increasingly popular as a software development model. It allows for valuable customer feedback and assists software designers and developers in understanding what is expected directly from sales.

The term “software prototyping” refers to the process of creating software application prototypes that demonstrate the functionality of the product under development but may not contain the exact logic of the original software.

3.1.1 Requirement Analysis

The project is started by analyzing the requirements and gathering them. During this phase, problem statements were identified along with objectives and requirements

for developing the system. Here, we briefly studied and collected various journals, articles, papers, and reports that were related to Parking Occupancy detection. We also compared various algorithms used in different sections to choose the right one.

3.1.2 System Design

In this phase, we modeled our system with the available elements. We parallelly learned a programming language and its frameworks and libraries such as Python, PyTorch, YOLO, and matplotlib that were required for building the system. System design is continued with system architecture, components, modules, interface, and data. We selected the algorithms needed for training purposes. We also designed various schema of the system component like Data Flow Diagram, etc.

3.1.3 Implementation

The system design is then implemented using programming languages like Python and different machine learning algorithms. In this two-step process, we used two YOLOv5 architecture for vehicle detection and RNN LSTM for speed estimation.

3.1.4 Testing

After the model was implemented, we tested the model using test data i.e. 20% of the collected data, and checked the output of the model.

3.1.5 Deployment

After the testing phase, the model was deployed in the local host.

3.2 Process Model

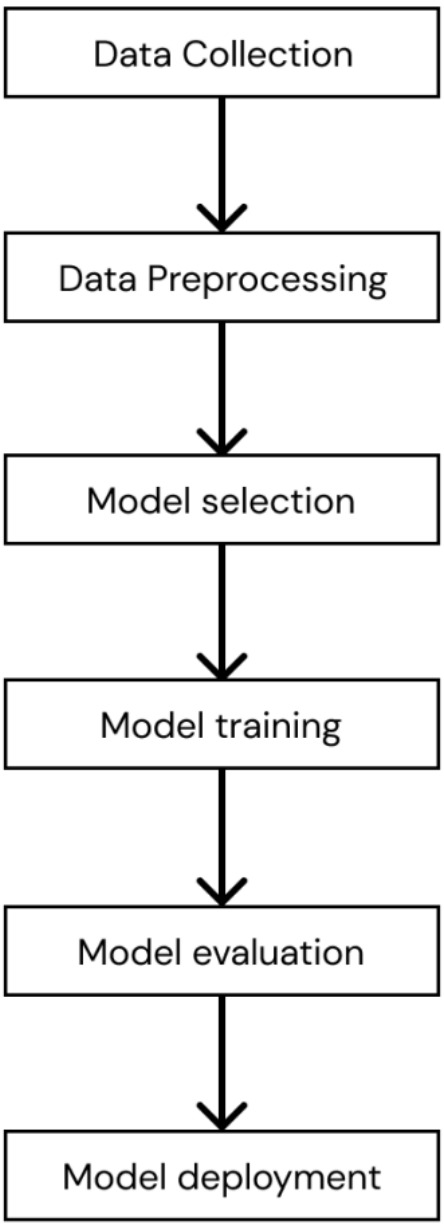


Figure 3. 2 Process Mode

The process model for parking occupancy detection using CNN typically involves the following steps:

1. Data Collection: The dataset used for this project involves images of parking lot spaces categorized into occupied and empty. This dataset was obtained from the paper “A dataset for audio-video based vehicle speed estimation” as mentioned in the literature review. The videos are 1920x1080 pixels, 30 fps and 10 seconds

each. Here is the distribution of data in each speed from the histogram from the paper.

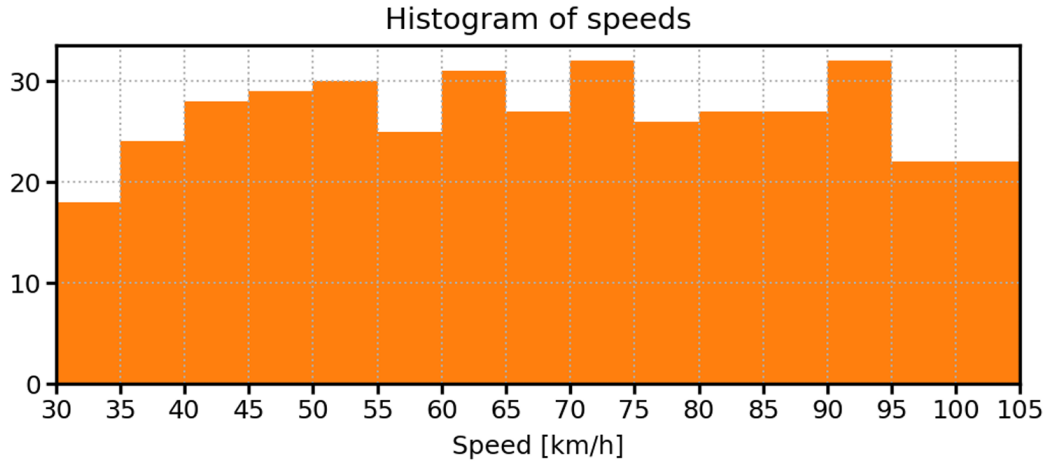


Figure 3. 3 Distribution of Data

As for YOLOv5 we used the pretrained model from Ultralytics, to detect cars in the VS13 dataset. Then we used the annotations provided by the YOLOv5 to fine tune the model to precisely detect cars in the frame.

Figure 3. 4 Pictures of training YOLOv5

2. Data Preprocessing: Since YOLOv5 from Ultralytics was used, the only preprocessing required for fine tuning model was to arrange them in the file system in the given format.

```
|--images
   |--train
         |-- img1.jpg
         |--….
   |--test
         |--img2.jpg
         |--…
|--labels
   |--train
         |-- img1.txt
         |--…

   |--test
         |-- img2.txt
         |--…
```

Figure 3. 5 File structure for training YOLOv5

The real preprocessing started only after the results from YOLOv5 were given. The YOLOv5 model gave output in form of bounding box coordinates. Initially the coordinates were recorded in the format of <frame no> which we added, and xyxy which was one of the outputs provided by v5.

After that, the .txt files were converted to a DataFrame (df) and the area for each bounding box was calculated. The df consists of car name with speed as index, frame no as column and area as value in each cell.

Any column with null values were dropped and only the columns with consecutive frames were kept. At the end, there were 80 columns from frame number 44 to frame number 123. And the difference between consecutive frame was calculated and stored in a new df.

	45	46	47	48	49	50	51	52	53	54	...	121	122	123
MercedesGLA_103	44.741	8.696	-10.646	47.200	36.748	5.660	28.596	53.313	3.781	31.705	...	1293.665	1373.890	928.054
MercedesGLA_55	148.432	74.957	199.548	141.091	255.101	53.614	149.877	221.423	115.699	48.916	...	78655.903	113730.629	137193.870
MercedesGLA_78	121.687	105.149	64.197	-52.560	89.798	55.124	42.380	142.763	71.403	98.569	...	19080.371	16702.894	24953.867
MercedesGLA_48	199.044	104.700	218.564	184.443	181.241	318.750	269.519	212.173	305.528	106.162	...	89176.337	-15968.512	16478.060
MercedesGLA_72	62.080	-3.694	42.424	52.229	43.591	85.025	69.558	54.654	93.670	80.226	...	4131.624	5507.454	4473.695

Figure 3. 6 Snippet of the Final Data Frame

3. Model selection: To build an effective machine learning model, it is essential to select an appropriate neural network architecture and training algorithm that aligns with the nature of the data and the specific task. In this project, we chose to utilize YOLOv5 architecture for car detection and RNN for Speed Estimation.

4. Model Training: In order to train a neural network, preprocessed data are utilized alongside an optimization algorithm such as Adam and loss function Root Mean Square Error. This helps to adjust the network's internal parameters, known as weights and biases, to minimize the error between predicted output and actual output. The optimization algorithm plays a vital role in the training process by facilitating the network to learn data patterns and adjusting its parameters accordingly. The same models were also trained on different hyperparameters namely learning rate with scheduler and no of epochs. By training models on preprocessed data, we accomplished high accuracy when it came to predicting speed estimation.

5. Model evaluation: The performance of the trained model is evaluated on a separate test dataset to assess its accuracy, and RMSE score. Any necessary adjustments to the model architecture or training process may be made based on the evaluation results.

   Overall, this process involves using deep learning techniques to develop a system for accurately identifying vehicles from camera and using it to estimate speed. The system has the potential to improve the accuracy and speed of speed estimation and to provide a valuable tool for the management of traffic.

CHAPTER 4: SYSTEM DESIGN

4.1 Block Diagram

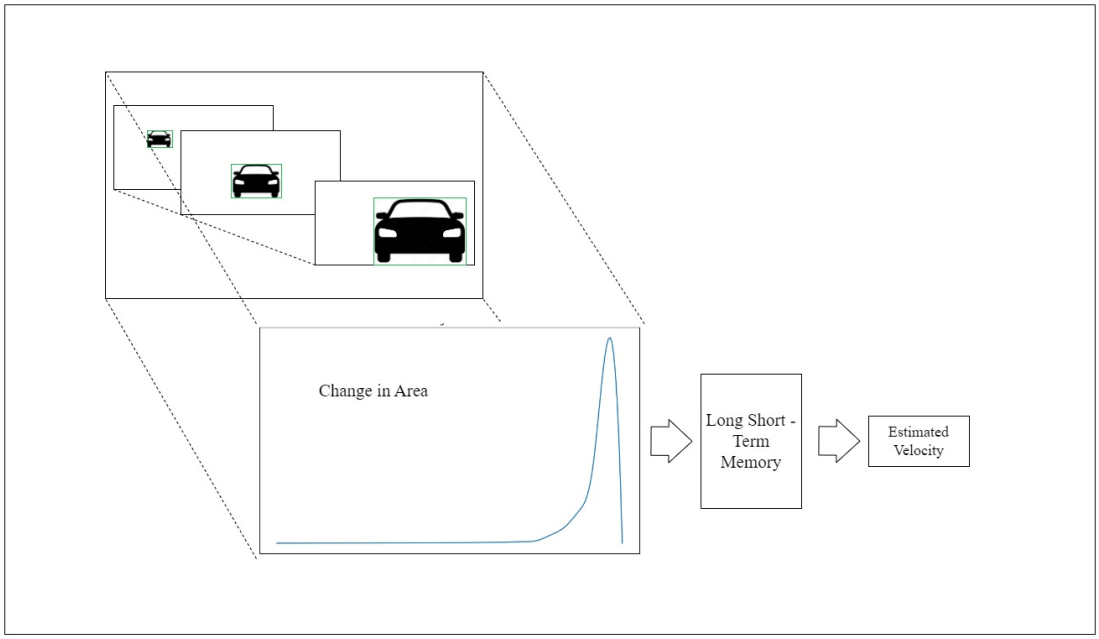


Figure 4. 1 Block Diagram of the system

The system is a two-stage system that initiates by detecting car in each frame and calculating the area of the bounding box and records it. Then it calculates the change in area and that data is sent to the LSTM for speed estimation.

4.2 Project Workflow

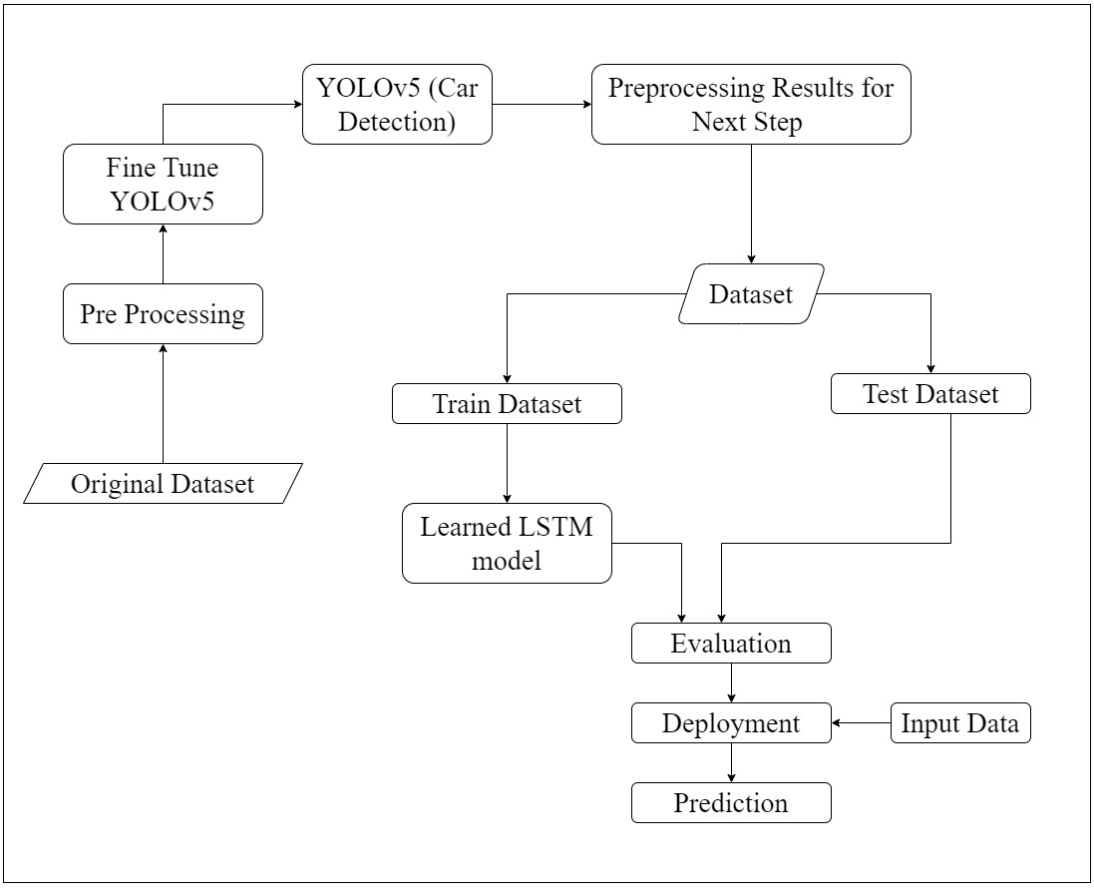


Figure 4. 2 Project Workflow

The workflow diagram outlines a comprehensive flow of the project while building a deep learning-based vehicle speed estimation system. It began with a YOLO model being fine-tuned with original dataset, VS13. Then, all the dataset was passed through model and bounding box area was obtained. Further processing was done to find the rate of difference. Then the dataset was divided into two parts test and train. Then a LSTM model was trained with the dataset, tested and deployed

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 Results

In this project, we aimed to estimate vehicle speed using a combination of YOLO (You Only Look Once) for object detection and LSTM (Long Short-Term Memory) networks for sequence learning, utilizing the VS13 dataset. The VS13 dataset consists of 400 videos, each containing a single vehicle, with 80 videos designated for testing. The speed estimation was conducted by measuring changes in the bounding box area provided by YOLO over time and feeding these measurements into an LSTM network.

Training Performance:

- During training, we observed a significant reduction in the Root Mean Square Error (RMSE). Initially, the RMSE was in the 60s, but it rapidly decreased, reaching approximately 10 within the first 300 epochs. After 600 epochs, the RMSE stabilized at around 4, indicating the model had effectively learned to estimate speed with high accuracy. The figure 1 shows the training curve.

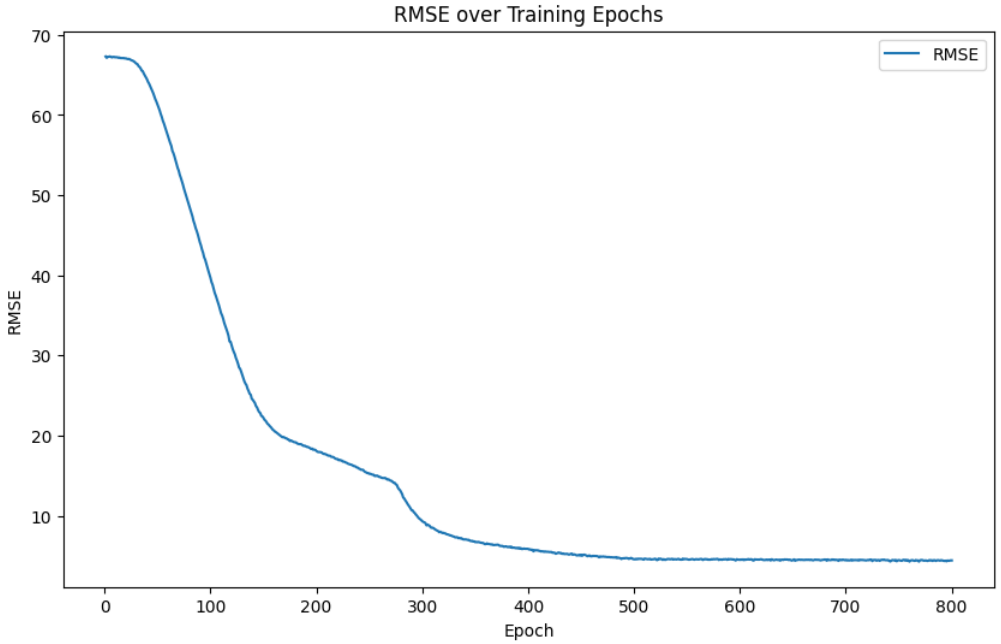


Figure 5. 1 Learning Curve

Testing Performance:

- On the test set, the model achieved a total RMSE of 8.1010. This result demonstrates that the model generalizes well to unseen data, maintaining a relatively low error rate in speed estimation.

Per Vehicle Performance:

The model's performance was further analyzed across different vehicle types within the dataset i.e. all 13 vehicles and 400 videos. The RMSE values for individual vehicles are listed in the table 2 below:

	Car	RMSE
1	Peugeot307	5.5319
2	RenaultCaptur	6.7812
3	Peugeot208	4.3728
4	NissanQashqai	5.3187
5	MercedesAMG550	4.0602
6	MercedesGLA	7.4322
7	CitroenC4Picasso	4.4368
8	KiaSportage	5.4585
9	RenaultScenic	7.6198
10	Peugeot3008	6.2481
11	OpelInsignia	6.0126
12	Mazda3	6.4512
13	VWPassat	5.5866
Average		5.7931

Table 5. 1 RMSE Across all the video

Here is the comprehensive graph for predicted vs actual speed.

5.2 Discussion

The results of our vehicle speed estimation project using YOLO and LSTM networks reveal several key insights into the performance and effectiveness of our approach.

Model Training and Performance:

- During the training phase, we observed a steep decline in RMSE from the 60s to approximately 10 within the first 300 epochs, eventually stabilizing around 4 after 600 epochs. This rapid decrease and eventual stabilization indicate that the model effectively learns to estimate speed as it is exposed to more data and training iterations.
- The final training RMSE of around 4% suggests a high degree of accuracy in the model's predictions by the end of the training process. It indicates that, on

average, the model's predictions deviate from the actual values by 4% of the range of the target variable.

Testing and Generalization:

- On the test set, the model achieved an RMSE of 8.1010%. While this is higher than the training RMSE, it still represents a reasonably low error rate, demonstrating that the model generalizes well to unseen data. This discrepancy between training and testing RMSE is typical in machine learning and suggests some overfitting, albeit not severe.
- The difference between training and testing RMSE highlights the importance of considering model generalization. While the model performs exceptionally well on the training data, the slightly higher RMSE on test data points to potential areas for improvement, such as regularization techniques or data augmentation to further enhance generalization.

Vehicle-Specific Performance:

- Analyzing RMSE across different vehicles revealed variability in the model's performance. The RMSE ranged from 4.0602% for the MercedesAMG550 to 7.6198% for the RenaultScenic, with an average RMSE of 5.7931% across all vehicles.
- This variability can be attributed to several factors, including differences in vehicle sizes, shapes, and speeds, as well as potential variations in video quality or environmental conditions. For example, larger or more uniquely shaped vehicles might result in more accurate bounding box predictions by YOLO, leading to better speed estimation.
- The highest RMSE for the RenaultScenic suggests that certain vehicles might present more challenges for the model. This could be due to their appearance or the nature of the videos in which they appear. Further investigation into these specific cases could provide insights for targeted model improvements.

Implications for Practical Use:

- The relatively low average RMSE of 5.7931% indicates that the model is effective for practical applications where moderate precision in speed estimation is acceptable. For applications requiring very high precision, additional refinement of the model might be necessary.

- The robustness of the approach across different vehicle types is promising for real-world deployment, suggesting that the model can handle a variety of vehicles without the need for significant adjustments or retraining.

CHAPTER 6: CONCLUSION

The project demonstrates the feasibility and effectiveness of using a combination of YOLO for object detection and LSTM for time-series analysis in estimating vehicle speed from video data. The significant reduction in RMSE during training and the reasonably low RMSE on the test set indicate that the model successfully learns and generalizes the task of speed estimation.

Despite some variability in performance across different vehicle types, the model consistently maintains a low average RMSE, highlighting its robustness and potential for practical applications. Future work could focus on addressing the observed variability and further improving the model's accuracy and generalization capabilities, possibly through enhanced preprocessing, additional data, or more sophisticated modeling techniques.

Overall, this project provides a strong foundation for vehicle speed estimation using deep learning techniques and paves the way for future research and development in this area.

CHAPTER 7: LIMITATIONS AND FUTURE ENHANCEMENTS

7.1 Limitations

- Single Vehicle Limitation: This model is designed to work with videos containing only a single vehicle in the frame. It cannot handle scenarios where multiple vehicles are present, which limits its applicability in real-world situations where traffic often involves multiple vehicles.
- Two-Step Estimation Process: The model operates as a two-step estimator. The initial step involves using YOLO to detect and track the vehicle, followed by a post-processing step to calculate the change in area, which is then fed into the LSTM network. This multi-step process can introduce complexities and potential points of failure.
- Lack of User Interface (UI): The current system does not include a user interface, making it less accessible for end-users who are not familiar with command-line operations or programming.
- Frame Requirement: The system requires at least 80 consecutive frames in which a car is detected to accurately estimate its speed. This requirement may not always be feasible, particularly in shorter video clips or videos with interruptions in vehicle detection.

7.2 Future Enhancements

For future enhancement, we can put our effort on:

- Handling Multiple Vehicles: Improving the model to work with multiple vehicles in the frame would significantly enhance its applicability in real-world traffic monitoring and analysis scenarios. This might involve using advanced tracking algorithms and more sophisticated scene understanding techniques.
- User-Friendly Interface: Developing a system with a user-friendly interface that includes limited and distinct features would make the tool more accessible to non-technical users. This could involve creating a graphical user interface (GUI) that simplifies the process of loading videos, running analyses, and viewing results.
- Algorithm Optimization: Further optimizing the underlying algorithms can improve the model's speed and accuracy. This could involve experimenting with different

configurations of YOLO and LSTM or integrating other deep learning architectures that may provide better performance.

- Reducing Frame Requirement: Reducing the number of consecutive frames needed for detection can make the model more flexible and usable in a wider range of scenarios. This might involve enhancing the model's robustness to shorter sequences or intermittent detections.
- Comparative Analysis with Other Algorithms: Exploring and comparing the performance of other algorithms could lead to improved speed estimation accuracy. For instance, integrating convolutional neural networks (CNNs) with LSTM, or using newer object detection frameworks like YOLOv8 or EfficientDet, might yield better results.
- Integration with Real-Time Systems: Implementing real-time processing capabilities can broaden the application of this model to live traffic monitoring systems. This would require optimizing the processing pipeline to handle video frames in real time without significant latency.

By addressing these limitations and pursuing these enhancements, the model can be made more robust, versatile, and user-friendly, thereby increasing its potential for practical deployment in various traffic monitoring and vehicle speed estimation applications.

REFERENCES

[1] K. Sangsuwan and Andmongkolekpanyapong, "Video-based vehicle speed estimation using speed measurement metrics," School of Engineering and Technology, Asian Institute of Technology, Khlong Nueng, Thailand, 2020.

[2] M. Won, "Intelligent traffic monitoring systems for vehicle classification: A survey," IEEE Access, vol. 8, pp. 73340-73358, 2020, doi: 10.1109/ACCESS.2020.2987634.

[3] D. F. Llorca, A. H. Martínez, and I. G. Daza, "Vision-based vehicle speed estimation: A survey," May 2021. [Online]. Available: https://www.example.com. [Accessed: May 21, 2021].

[4] D. Bell, W. Xiao, and P. James, "Accurate vehicle speed estimation from monocular camera footage," in ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Aug. 2020.

[5] A. Cvijetić, S. Djukanović, and A. Peruničić, "Deep learning-based vehicle speed estimation using the YOLO detector and 1D-CNN," in Proc. 27th Int. Conf. Information Technology (IT), Zabljak, Montenegro, 2023, pp. 1-4, doi: 10.1109/IT57431.2023.10078518.

[6] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," Dec. 2017. [Online]. Available: https://arxiv.org/abs/1801.01078. [Accessed: Feb. 22, 2018].

[7] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157-166, Mar. 1994, doi: 10.1109/72.279181.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[9] G. Jocher, "YOLOv5 by Ultralytics," 2020. [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed: May 2024].

[10] M. Horvat and G. Gledec, "A comparative study of YOLOv5 models performance for image localization and classification," Dept. of Appl. Comput., Univ. Zagreb, Zagreb, Croatia, 2020. [Online]. Available:

https://github.com/mhorvat/YOLOv5-models-comparison. [Accessed: May 2024].

[11] S. Djukanović, N. Bulatović, and I. Čavor, "A dataset for audio-video based vehicle speed estimation," in Proc. 30th Telecommun. Forum (TELFOR), Belgrade, Serbia, 2022, pp. 1-4, doi: 10.1109/TELFOR56187.2022.9983773.

Appendices

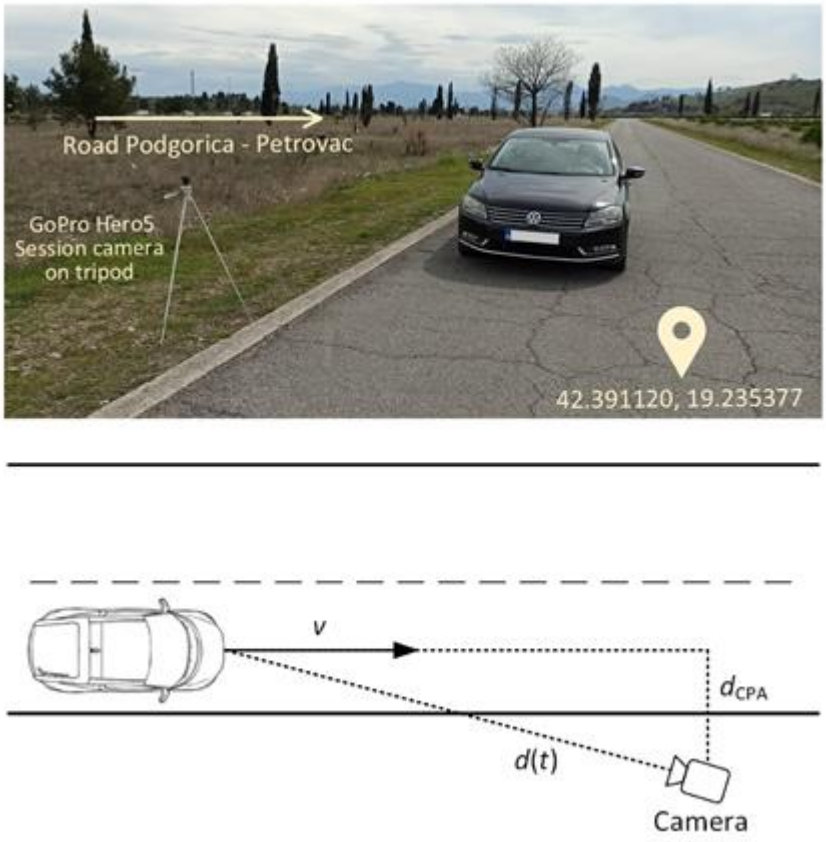


Figure A Recording Setup of the dataset

```python
# Model definition
class SpeedEstimatorRNN(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, output_size):
        super(SpeedEstimatorRNN, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc1 = nn.Linear(hidden_size, 64)
        self.fc2 = nn.Linear(64, output_size)

    def forward(self, x):
        out, _ = self.lstm(x)
        out = out[:, -1, :]  # take the last output of the LSTM
        out = nn.ReLU()(self.fc1(out))
        out = self.fc2(out)
        return out
```

Figure B LSTM Model

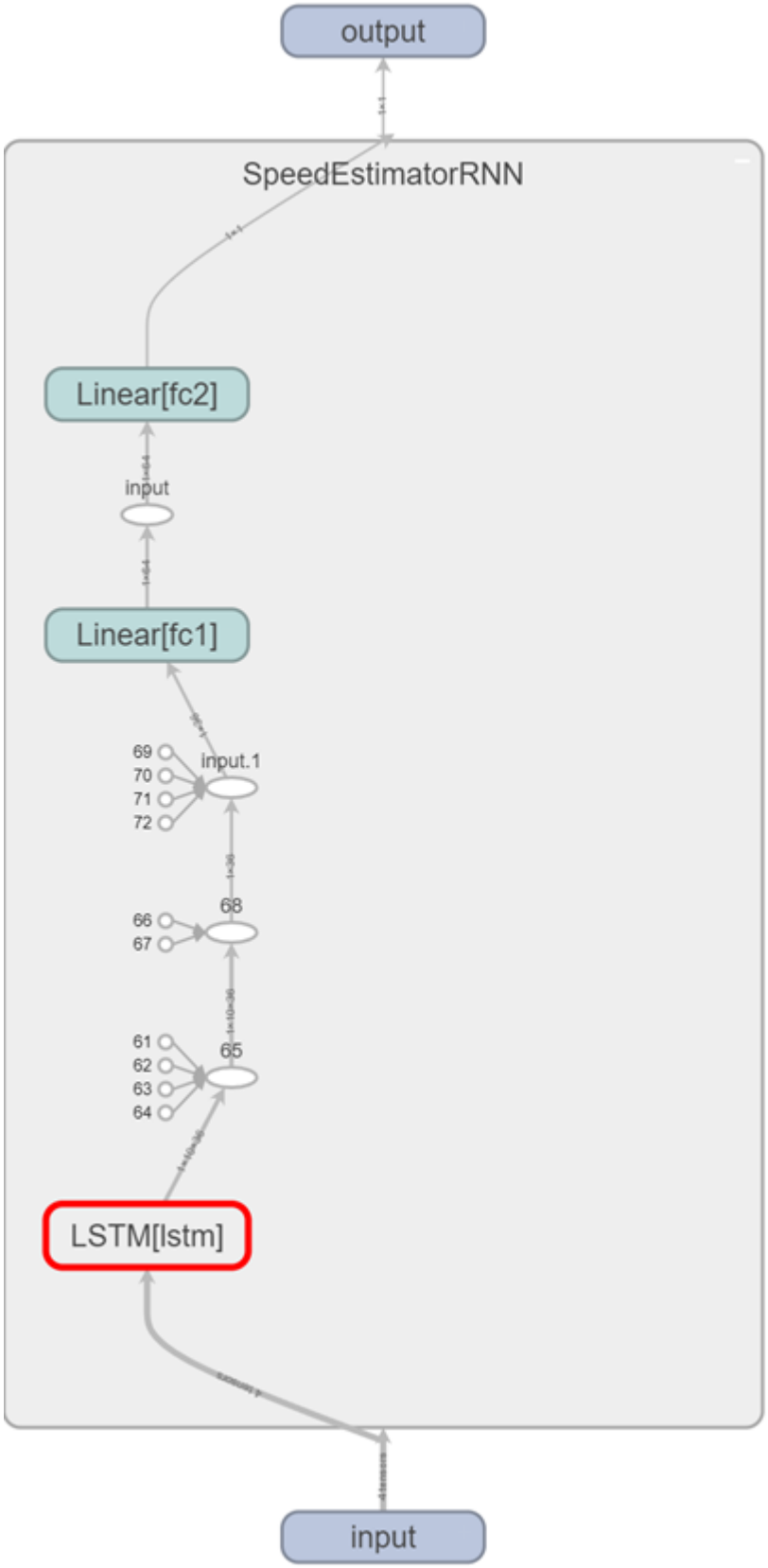
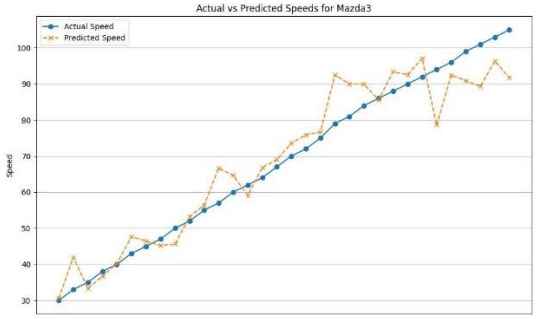
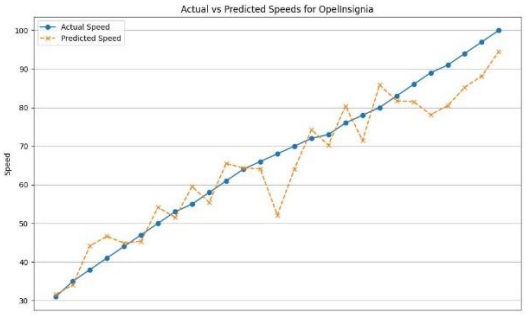
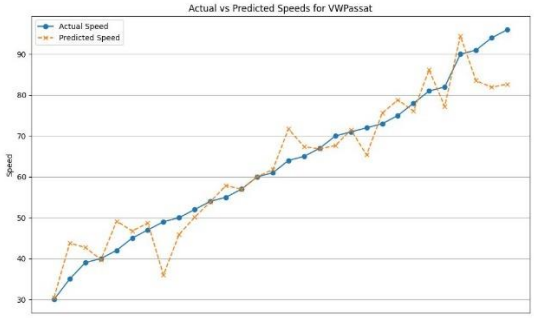
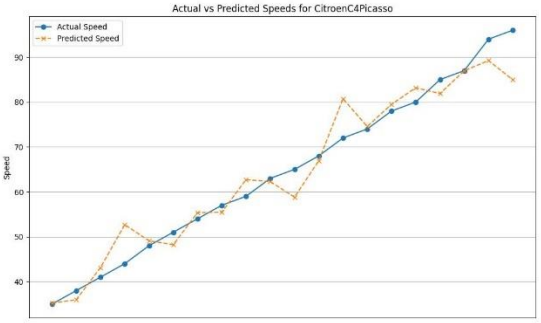


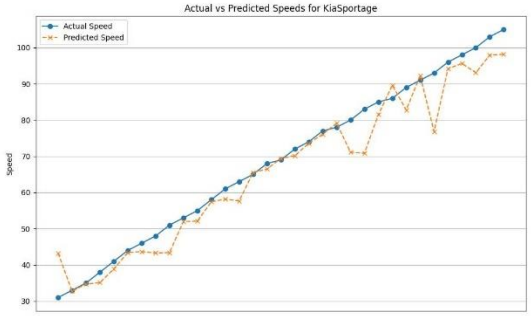
Figure C LSTM Model by TensorBoard

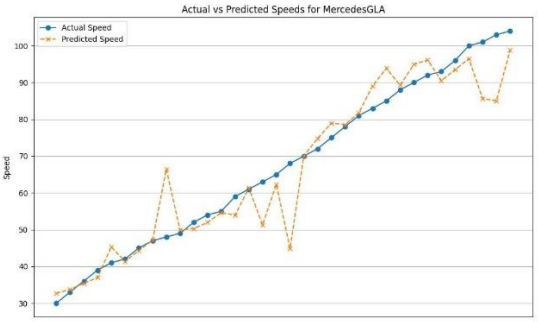

Actual vs Predicted Speeds for Mazda3
Actual Speed
Predicted Speed
Speed

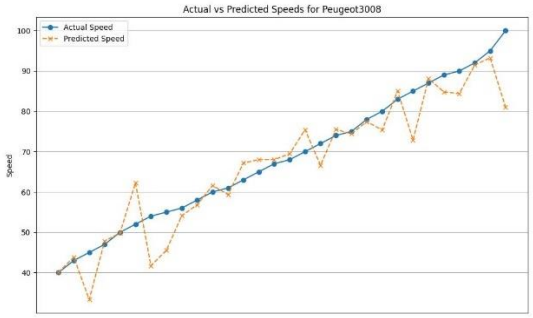

Actual vs Predicted Speeds for OpelInsignia
Actual Speed
Predicted Speed
Speed

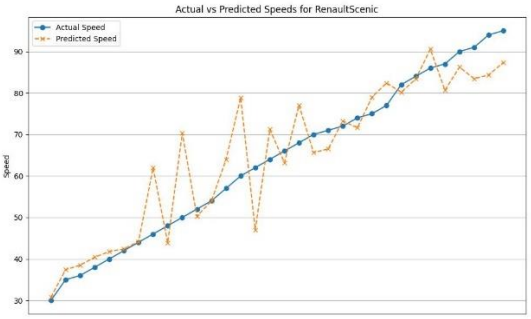

Actual vs Predicted Speeds for VWPassat
Actual Speed
Predicted Speed
Speed

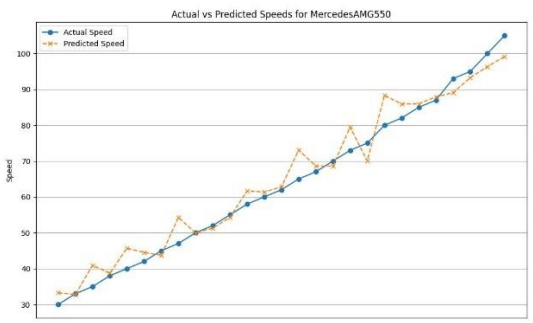

Actual vs Predicted Speeds for CitroenC4Picasso
Actual Speed
Predicted Speed
Speed

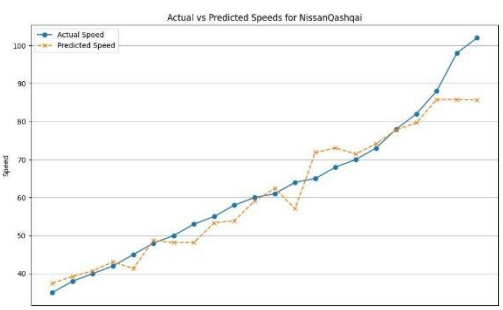

Actual vs Predicted Speeds for KiaSportage
Actual Speed
Predicted Speed
Speed


Actual vs Predicted Speeds for MercedesGLA
Actual Speed
Predicted Speed
Speed


Actual vs Predicted Speeds for Peugeot3008
Actual Speed
Predicted Speed
Speed


Actual vs Predicted Speeds for RenaultScenic
Actual Speed
Predicted Speed
Speed


Actual vs Predicted Speeds for MercedesAMG550
Actual Speed
Predicted Speed
Speed


Actual vs Predicted Speeds for NissanQashqai
Actual Speed
Predicted Speed
Speed

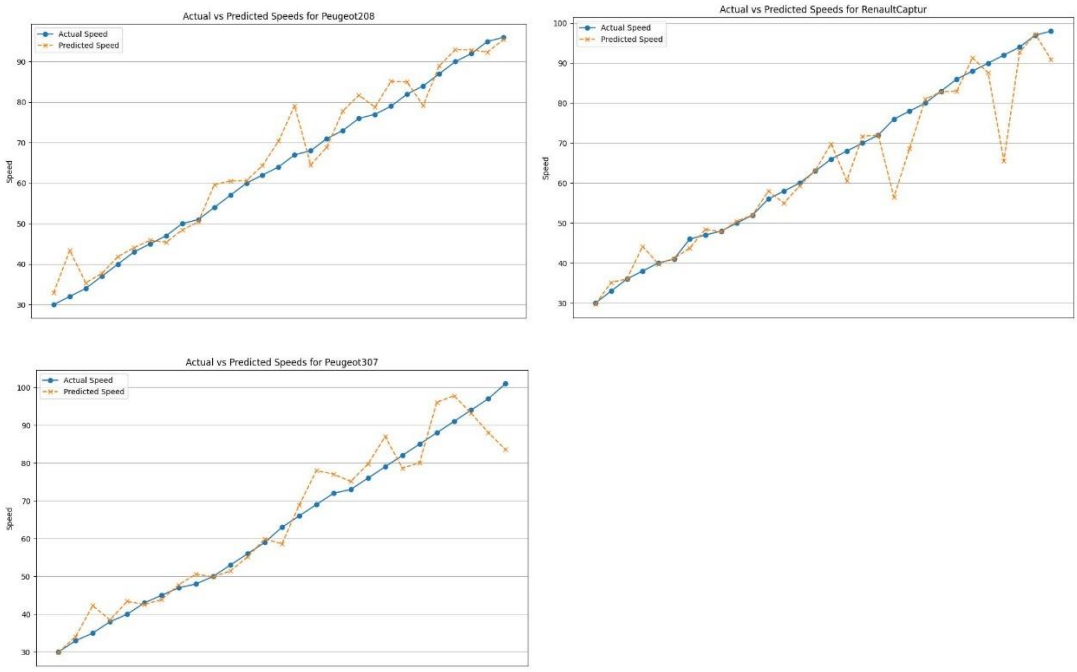
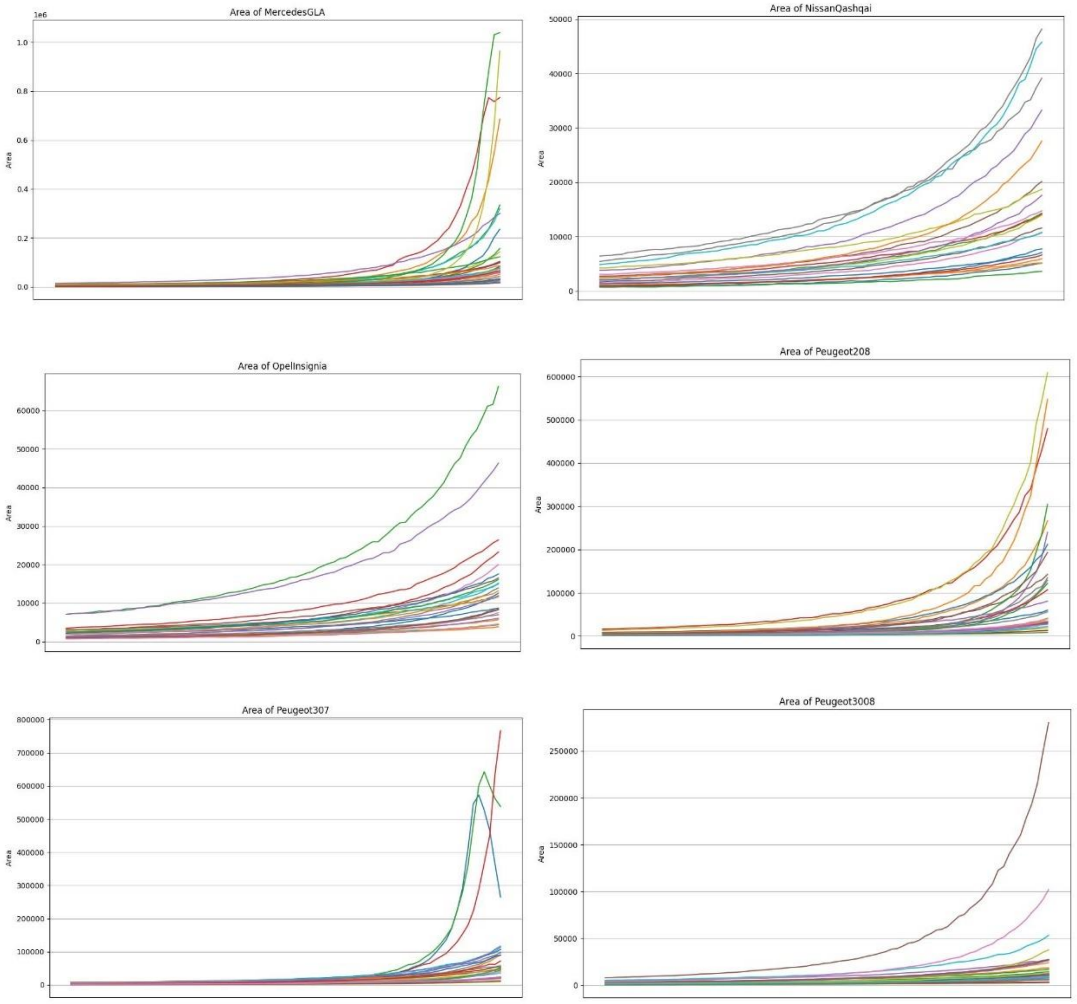


Figure D Actual vs Predicted Speed Graph



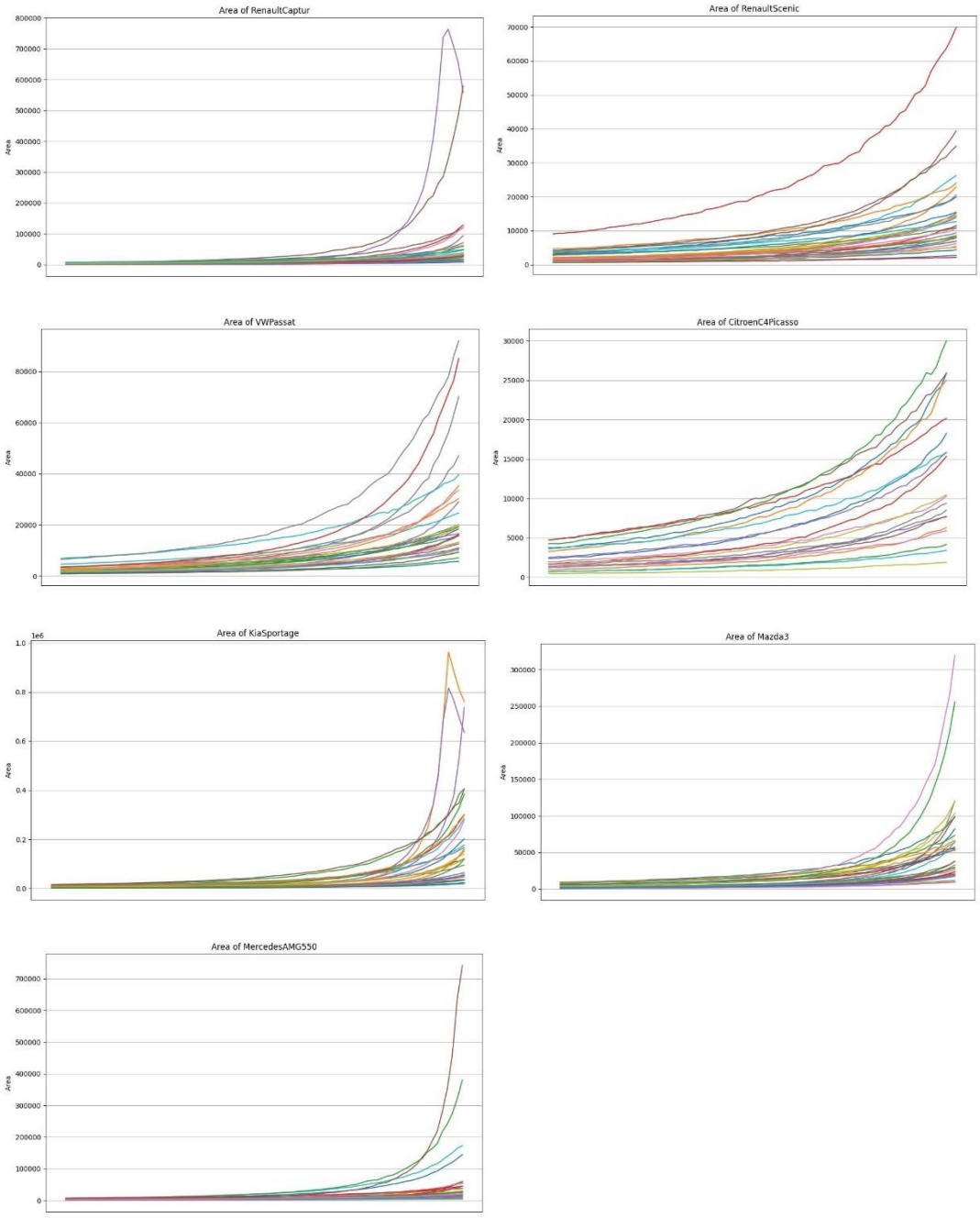
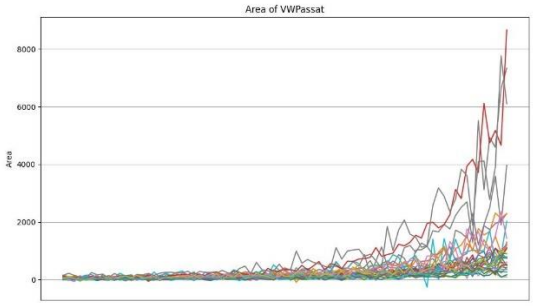
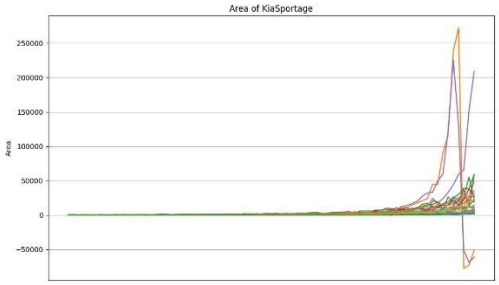
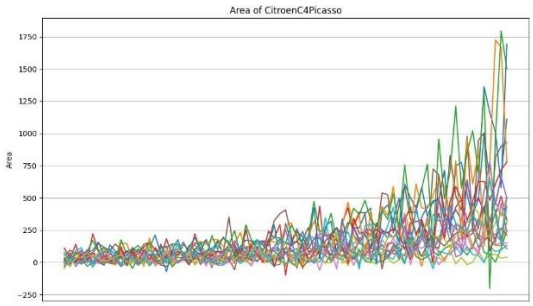
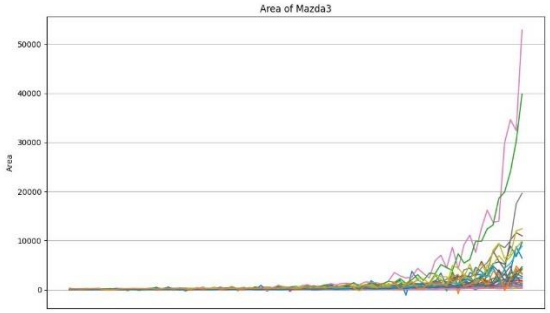


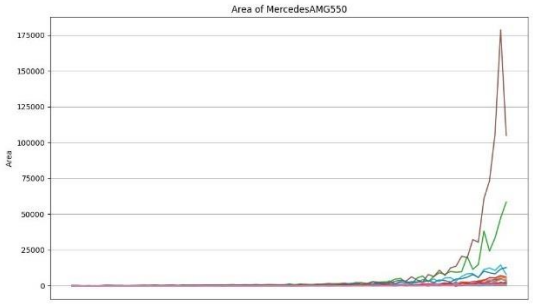
Figure E Area Vs Frame

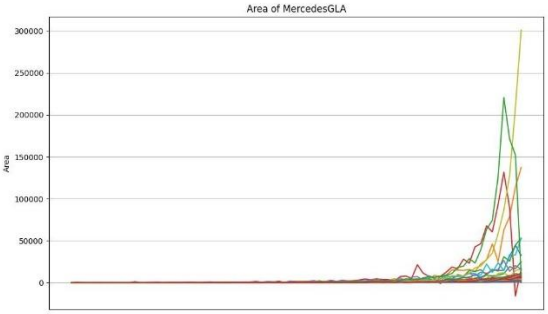

Area of VWPassat
Area
8000
6000
4000
2000
0

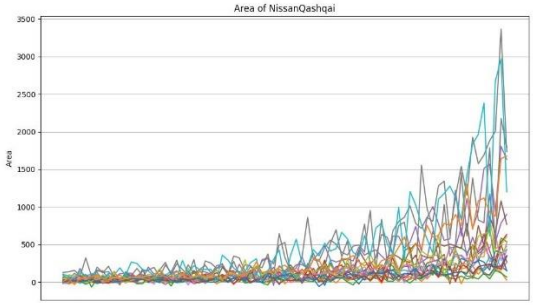

Area of KiaSportage
Area
250000
200000
150000
100000
50000
0
−50000

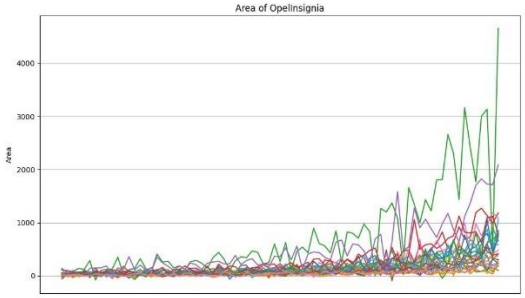

Area of CitroenC4Picasso
Area
1750
1500
1250
1000
750
500
250
0
−250

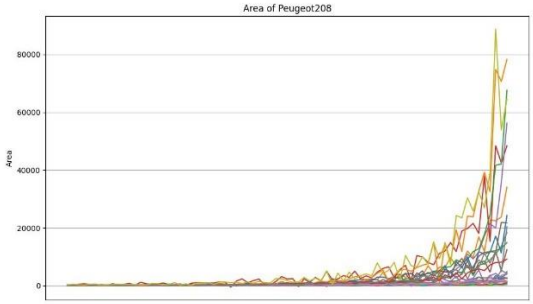

Area of Mazda3
Area
50000
40000
30000
20000
10000
0

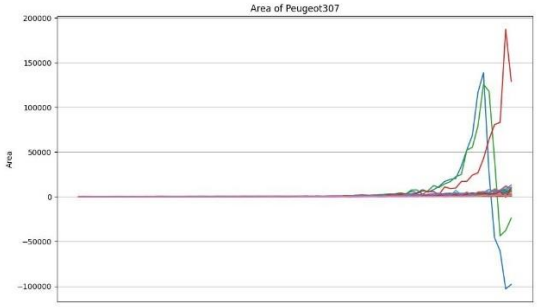

Area of MercedesAMG550
Area
175000
150000
125000
100000
75000
50000
25000
0


Area of MercedesGLA
Area
300000
250000
200000
150000
100000
50000
0


Area of NissanQashqai
Area
3500
3000
2500
2000
1500
1000
500
0


Area of OpelInsignia
Area
4000
3000
2000
1000
0


Area of Peugeot208
Area
80000
60000
40000
20000
0


Area of Peugeot307
Area
200000
150000
100000
50000
0
−50000
−100000

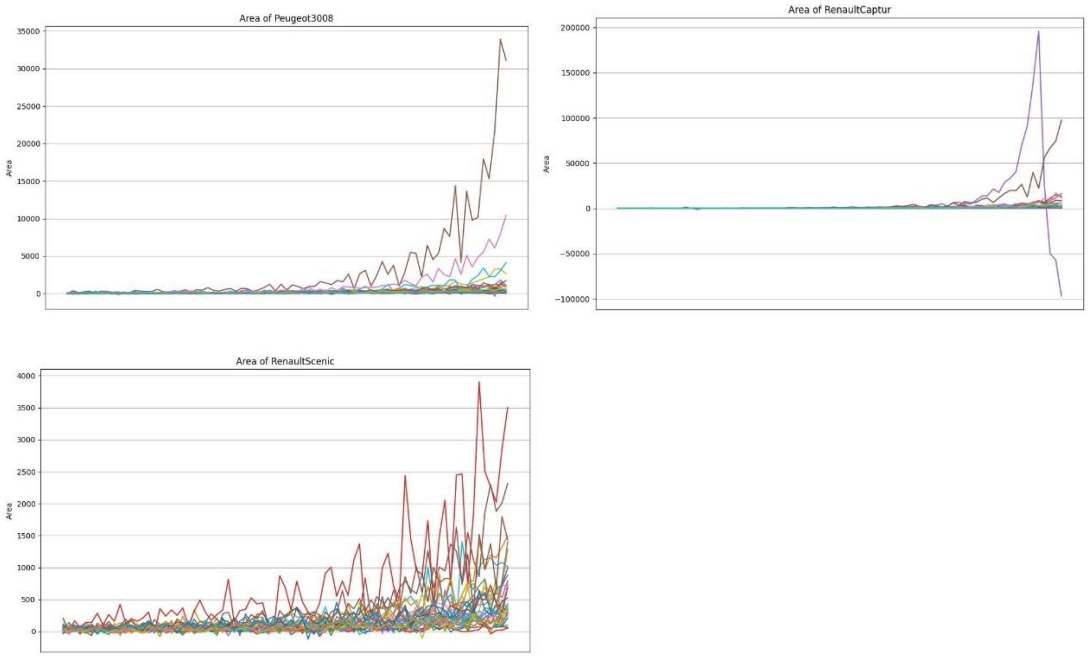


Figure F Change in Area vs Frame