

Trajectory Prediction of Road Users for Autonomous Vehicles



Author

Muhammad Siddiqui

00000318300

Supervisor

Dr. Hamid Jabbar


MASTERS IN MECHATRONICS ENGINEERING


**COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING,
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD, PAKISTAN.**


(August 2023)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis by NS Muhammd Siddiqui Registration No. 00000318300, of Electrical and Mechanical Engineering College has been vetted by undersigned, found complete in all respects as per NUST Statues/Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature:  _____
Name of Supervisor: Hamid Jabbar
Dated: 15 - 08 - 2023

Signature of HOD:  _____
Dr. Amir Hamza
Date: 15 - 08 - 2023

Signature of Dean:  _____
Brig Dr. Nasir Rashid
Date: 15 AUG 2023

I dedicate this thesis to Alan Turing whose imaginary machine acted as the foundation of all modern computing and thus making Deep Neural Networks possible, and to my supervisor, Dr. Hamid Jabbar, without the kind and patient guidance of whom I would not have been able to do this thesis work and to my beloved wife without the support and understanding of whom it had never been possible.

Acknowledgments

I am deeply thankful to my supervisor, Dr. Hamid Jabbar for his kind, helpful and technical input. I feel no doubt in saying that it was he because of whom I am able to complete this thesis work.

I am also very grateful to Dr. Ali Hassan. His understanding, accommodating, expert and knowledgeable guidance has made me through this.

It is also not justice if I don't pay my thanks and gratitude towards my guidance examination committee member Dr. Umar Shahbaz whose constructive criticism enabled me to see and thus fulfill my blind spots.

The guidance from my another guidance examination committee member, Dr. M. Osama Ali was also extra ordinary and worth paying the gratitude for.

I would like to thank Dr. Tahir Habib Nawaz for his input, time and effort he invested in me, and it acted as the guiding torch for this whole thesis work.

I would also like to pay my heartiest and warm thanks, gratitude and acknowledgement to Dr. Hassan Elahi. His feedback, encouragement and knowledge acted as a constant fuel for this whole journey.

It is never be completed if I don't mention and acknowledge the technical discussions I had with Mr. Fahad Mattoo were precious and somewhat acted as primary factors in the success of this thesis work.

Abstract

Trajectory prediction of road users is a very important task in autonomous vehicles for road safety. It becomes even more important when road users are pedestrians being more vulnerable. Recently, machine learning based approaches have been used for trajectory prediction of pedestrians including deep learning using deep neural networks. It is a question that what are the best features to include in order to predict the future trajectory of pedestrians more accurately. Moreover, the recurrent neural networks (RNNs) based decoder in encoder-decoder architecture suffer from the accumulated error that degrades long term predictions. In this work, feature selection for the trajectory prediction of pedestrians has been done and a novel bi-directional RNN decoder has been proposed. JAAD and PIE datasets have been used which focus on pedestrians. Different groups of available features in the datasets were selected, and the exploration analysis was done. Furthermore, in order to address the issue of accumulated error in uni-directional RNN decoders, a bi-directional decoder which has a forward RNN and a backward RNN, was proposed and compared with the uni-directional decoder. Results show that attributes features when added with spatial features improve the accuracy of the trajectory prediction and the proposed bi-directional decoder improves the accuracy of long term trajectory prediction.

Keywords: *Autonomous Vehicles, Machine Learning, Deep Learning, Deep Neural Networks, Trajectory Prediction, Recurrent Neural Networks*

Table of Contents

Acknowledgments	IV
Abstract	VI
Table of Contents	VII
List of Figures.....	XII
List of Tables	XV
1 Introduction	1
1.1 Background and Motivation	1
1.2 Levels of Automation.....	3
1.2.1 Level 0: No Automation (Traditional Human-Only Driving)	4
1.2.2 Level 1: Driver Assistance (Basic Automation)	4
1.2.3 Level 2: Partial Automation (Combined Automation)	4
1.2.4 Level 3: Conditional Automation (Limited Self-Driving)	5
1.2.5 Level 4: High Automation (Full Self-Driving in Limited Circumstances)	5
1.2.6 Level 5: Full Automation (True Self-Driving).....	5
1.3 Modules of Autonomous Vehicles.....	6
1.3.1 Sensing Module	6

1.3.1.1	Camera	6
1.3.1.2	LiDAR	7
1.3.1.3	RADAR.....	7
1.3.1.4	Ultrasonic.....	8
1.3.1.5	IMU	8
1.3.2	Perception Module	11
1.3.2.1	Object Detection and Classification	11
1.3.2.2	Semantic Segmentation	12
1.3.2.3	Environmental Mapping and Localization.....	12
1.3.3	Prediction Module	12
1.3.3.1	Trajectory prediction	13
1.3.4	Planning Module	14
1.3.5	Control Module	14
1.4	Deep Neural Networks.....	15
1.4.1	The Working of Deep Neural Networks	15
1.4.1.1	Architecture:.....	15
1.4.1.2	Feedforward Process:.....	15
1.4.1.3	Backpropagation:.....	15
1.4.2	Types of Deep Neural Networks	16
1.4.2.1	Convolutional Neural Networks (CNNs).....	16
1.4.2.2	Recurrent Neural Networks (RNNs)	16
1.4.2.3	Autoencoders	16
1.4.2.4	Generative Adversarial Networks (GANs).....	16
1.5	Recurrent Neural Networks	16

1.5.1	Forward Propagation	17
1.5.2	Backpropagation Through Time (BPTT)	17
1.6	The Specific Task Targeted by This Thesis Work.....	18
1.7	Problem Formulation	18
1.8	Research Objectives	19
2	Literature Review	20
2.1	Classification of Methods of Trajectory Prediction	20
2.1.1	Physics-Based	20
2.1.2	Learning Based.....	21
2.2	Unimodal vs Multimodal Trajectory Prediction	21
2.2.1	Unimodal Trajectory Prediction.....	21
2.2.2	Multimodal Trajectory Prediction.....	21
2.3	Multi-Agent vs. Single-Agent Trajectory Prediction	22
2.3.1	Multi-Agent Trajectory Prediction.....	22
2.3.2	Single-Agent Trajectory Prediction.....	22
2.4	Context Awareness.....	23
2.5	Goal Driven/Conditioned.....	23
2.5.1	End Goal.....	24
2.5.2	Grid Based.....	24
2.5.3	Stepwise Goals.....	24
2.6	Datasets.....	25
2.6.1	Sensors-based Classification	25
2.6.2	Type of Target-based Classification.....	26
2.6.3	Point of View-based Classification.....	26

2.6.3.1	First Person View (FPV).....	26
2.6.3.2	Bird’s Eye View (BEV).....	26
2.6.4	Driving Environment-based Classification.....	27
2.6.4.1	Highways.....	27
2.6.4.2	Rural or Urban Driving Environment.....	27
2.6.4.3	Time of the Day.....	27
2.6.4.4	Weather.....	28
3	Methodology.....	30
3.1	Datasets Seclction.....	30
3.1.1	JAAD Dataset.....	30
3.1.2	PIE.....	32
3.2	Baseline Method.....	33
3.3	Model Architecture.....	35
3.3.1	Data Generation.....	35
3.3.2	Encoder.....	37
3.3.3	Stepwise Goal Estimator (SGE).....	39
3.3.4	Conditional Variational Autoencoder (CVAE).....	41
3.3.5	Bi-directional Decoder.....	42
3.3.6	Loss Function.....	47
3.4	Evaluation Metrics.....	48
3.4.1	Mean Square Error (MSE).....	48
3.4.2	Center Mean Square Error (C_{MSE}).....	48
3.4.3	Center Final Mean Square Error (CF_{MSE}).....	48
4	Experimentation.....	49

4.1	Implementation Details.....	49
4.2	Machine Setup	49
4.3	Feature Selection Experiments	50
4.3.1	Exploration Analysis	50
4.3.2	Comparative Analysis.....	51
4.4	Bi-Directional RNN Decoder Experiments.....	51
5	Results and Discussion.....	52
5.1	Reproduced Results of Baseline Method	52
5.2	Feature Selection.....	53
5.2.1	Exploration Analysis	53
5.2.2	Comparative Analysis.....	55
5.3	Bi-Directional RNN Decoder.....	56
5.4	Qualitative Results	58
6	Conclusion and Future Work.....	60
6.1	Conclusion	60
6.2	Future Work.....	61
	References	XVII
	Thesis Completion Certificate.....	XXV

List of Figures

1.1	Levels of automation [1].....	3
1.2	Illustration of the task of trajectory prediction of pedestrians for autonomous vehicles. Yellow represents observed trajectory, Red represents future ground truth trajectory, Green represents predicted trajectory. (a) Graphical illustration. (b) Symbolic illustration.....	18
1.3	Graphical abstract, which depicts this work in a concise pictorial form. 1st Column (left) illustrates the task of trajectory prediction of pedestrians for autonomous vehicles. 2nd Column (middle) illustrates the research gaps found and targeted in this work and methodology. 3rd Column (right) illustrates the conclusion.	19
3.1	Block diagram of the model architecture. All blocks are explained in detail in the later sections. All inputs and outputs are labeled and can be matched. (1) Encoder, detailed diagram in Figure 3.2 (2) Stepwise goal estimator (SGE), detailed diagram in Figure 3.3 (3) Conditional Variational Autoencoder (CVAE), detailed diagram in Figure 3.4 (4) Bi-directional Decoder, detailed diagram in Figure 3.5.....	36
3.2	Detailed diagram of the encoder module. All inputs and outputs match with the block diagram shown in Figure 3.1.....	38
3.3	Detailed diagram of the stepwise goal estimator (SGE) module. All inputs and outputs match with the block diagram shown in Figure 3.1	39

3.4	Detailed diagram of the conditional variational autoencoder (CVAE) module. All inputs and outputs match with the block diagram shown in Figure 3.1.....	41
3.5	Detailed diagram of the bi-directional decoder module. All inputs and outputs match with the block diagram shown in Figure 3.1	42
3.6	Symbolic working of bi-directional RNN. The ellipses represent the accumulated error. Forward pass is shown in green, and are increasing from time step $t + 1$ to $t + l_p$. Backward pass is shown in blue, and it is greater for time step $t + 1$ to $t + l_p$	45
4.1	Hardware and software stack used for the experiments.....	51
5.1	Line graph comparing the results of different sets of selected features on <i>JAAD</i> and <i>PIE</i> datasets. Only the evaluation metric <i>MSE</i> for prediction horizon 0.5s, 1.0s and 1.5s is shown. Bounding box is present in all the sets of selected features.....	54
5.2	Bar graph comparing the results of different sets of selected features on <i>JAAD</i> and <i>PIE</i> datasets. Evaluation metric C_{MSE} and CF_{MSE} for prediction horizon of 1.5s is shown. Bounding box is present in all the sets of selected features.	55
5.3	Graph comparing the results of the best selected features on <i>JAAD</i> and <i>PIE</i> datasets with reproduced results of the baseline method SGNet-ED [2].....	56
5.4	Graph comparing the results of the proposed method Bi-SGNet with the reproduced results of the baseline method SGNet-ED [2] on <i>JAAD</i> and <i>PIE</i> datasets. This shows that the proposed method outperforms the baseline method in long term trajectory prediction of pedestrians on both datasets.....	58

5.5	Qualitative results of the proposed method on <i>JAAD</i> and <i>PIE</i> datasets. (a) <i>JAAD</i> dataset, (b) <i>PIE</i> dataset. Yellow represents observed trajectory, Red represents future ground truth trajectory, Blue represents predicted trajectory by the baseline method, Green represents predicted trajectory by the proposed method of bi-directional <i>RNN</i> decoder (<i>Bi – SNet</i>).....	59
6.1	Failure cases. Yellow represents observed trajectory, Red represents the ground truth trajectory, Green represents predicted trajectory. (a) Failure because of lack of interaction information. (b) Failure because of lack of roads and lanes information.	61

List of Tables

1.1	Comparison of the Sensors Used in AVs.....	10
2.1	Literature Review of Methods	25
2.2	Literature Review of Datasets	29
3.1	Selected Datasets	30
3.2	<i>JAAD</i> Dataset Camera Sensor Details.....	31
3.3	<i>JAAD</i> Dataset Details	31
3.4	<i>PIE</i> Dataset Camera Sensor Details	32
3.5	<i>PIE</i> Dataset [3] Sets of Video Clips.....	32
3.6	<i>PIE</i> Dataset [3] Details	33
3.7	Classification of the Baseline Method SGNet-ED [2]	34
3.8	Comparison of the Baseline Method SGNet-ED [2] with other State-of-the-Art Methods on <i>JAAD</i> [4] and <i>PIE</i> [3] Datasets (↓ <i>lower the better</i>).....	34
4.1	Machine Setup (<i>hardware + software</i>).....	50
5.1	Reproduced Results of the Baseline Method on the Test Bench (↓ <i>lower the better</i>).....	52
5.2	Comparison of Selected Features (↓ <i>lower the better</i>)	54

5.3	Comparison of the Best Selected Features with the Reproduced Results of the Baseline Method SGNet-ED [2] on <i>JAAD</i> and <i>PIE</i> Datasets (<i>↓ lower the better</i>).....	56
5.4	Comparison of the Proposed Bi-Directional Decoder Bi-SGNet, with the Reproduced Results of the Baseline Method SGNet-ED [2] on <i>JAAD</i> and <i>PIE</i> Datasets (<i>↓ lower the better</i>).....	57

Chapter 1: Introduction

1.1 Background and Motivation

The concept of self-driving vehicles is not entirely new; it has been a subject of fascination and speculation for decades. Starting in the 1920s [5], it took place in the 1980s when researchers were able to develop some automated highway systems [6], [7]. However, recent advancements in artificial intelligence, machine learning, sensor technology, and data processing have catapulted autonomous vehicles from a theoretical possibility to a tangible reality. As a result, major automotive manufacturers, tech giants, and startups alike have invested considerable resources in the development of autonomous driving systems, leading to significant progress in the field.

The need for autonomous vehicles is multi-faceted and encompasses various societal, environmental, and economic factors. Foremost among these is the alarming toll of road traffic accidents, which claim millions of lives each year and cause substantial economic losses. Human error remains the primary cause of most accidents, and autonomous vehicles, with their ability to operate with precision and without fatigue, hold the potential to drastically reduce the number of collisions and fatalities on our roadways.

Furthermore, autonomous vehicles offer a compelling solution to alleviate traffic congestion, a perennial issue plaguing densely populated urban centers. By optimizing routes, adhering to efficient driving patterns, and facilitating seamless coordination between vehicles, these self-driving systems can significantly enhance traffic flow and overall transportation efficiency. Consequently, this would lead to reduced travel times, lower fuel consumption, and decreased emissions,

contributing to a more sustainable and environmentally friendly transportation network [5].

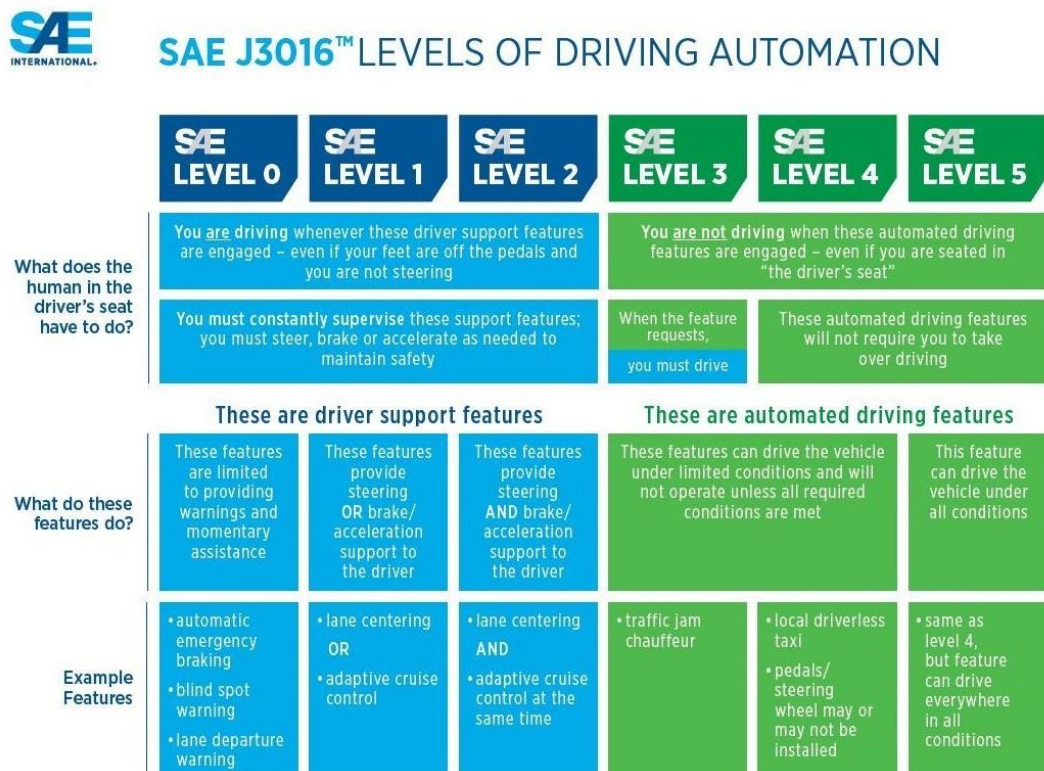
Beyond the individual benefits, the integration of autonomous vehicles has the potential to transform our cities and urban infrastructure. By fostering the development of smart cities that are interconnected and optimized for autonomous transportation, we can unlock new possibilities for urban planning, land use [5], and public space utilization. It can even lead to parking problems. Shoup [8] has done some estimations that on average about 31

It can play a contribution to environmental challenges. The transportation sector is a significant contributor to greenhouse gas emissions and air pollution. It is estimated that transport is the cause of about one quarter to two third emissions of greenhouse gasses [9]. By introducing autonomous vehicles into the transportation mix, we can potentially reduce the environmental impact of driving. Self-driving cars can be programmed to drive more efficiently, minimizing fuel consumption and emissions. Moreover, the growing interest in electric and autonomous vehicles can create an opportunity for synergies, promoting the adoption of electric, autonomous fleets, which would further contribute to reducing carbon emissions.

The journey from human drivers to fully autonomous vehicles is not possible. It must go through a phase where for quite a long time, the roads will be shared by both human drivers and autonomous vehicles. For this, we must integrate intelligence into the autonomous vehicle systems which can understand and predict human behaviors and predict their future trajectories. So that they can safely cooperate on the same roads with human drivers.

1.2 Levels of Automation

The automotive industry is currently witnessing a paradigm shift with the advent of autonomous driving technologies. As vehicles become more intelligent and capable of self-driving, it is crucial to have a standardized framework to classify the varying degrees of automation. The Society of Automotive Engineers (SAE) has taken up this challenge by introducing the SAE Levels of Automated Driving, a comprehensive categorization system that offers a clear understanding of the capabilities and limitations of autonomous vehicles. In this section, we will delve into the five SAE levels, each representing a step towards full autonomy, and explore the implications of these advancements on our roadways.



The figure is a matrix titled "SAE J3016™ LEVELS OF DRIVING AUTOMATION" with the SAE International logo. It is organized into columns for SAE Level 0 through SAE Level 5. The rows are categorized by questions: "What does the human in the driver's seat have to do?", "What do these features do?", and "Example Features". The matrix is color-coded: Level 0-2 are blue, Level 3 is light green, and Level 4-5 are dark green.

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
What do these features do?	These are driver support features			These are automated driving features		
	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/acceleration support to the driver	These features provide steering AND brake/acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met	This feature can drive the vehicle under all conditions	
Example Features	<ul style="list-style-type: none"> • automatic emergency braking • blind spot warning • lane departure warning 	<ul style="list-style-type: none"> • lane centering OR • adaptive cruise control 	<ul style="list-style-type: none"> • lane centering AND • adaptive cruise control at the same time 	<ul style="list-style-type: none"> • traffic jam chauffeur 	<ul style="list-style-type: none"> • local driverless taxi • pedals/steering wheel may or may not be installed 	<ul style="list-style-type: none"> • same as level 4, but feature can drive everywhere in all conditions

Figure 1.1: Levels of automation [1]

1.2.1 Level 0: No Automation (Traditional Human-Only Driving)

At Level 0, there is no automation involved, and the human driver is solely responsible for all aspects of vehicle operation. In this traditional driving scenario, the car provides no assistance or automation features, requiring the driver to handle every task, from acceleration and steering to braking and monitoring the environment. As we progress from Level 0 to higher levels, the introduction of automation gradually eases the burden on the driver and enhances overall safety.

1.2.2 Level 1: Driver Assistance (Basic Automation)

Level 1 represents the introduction of basic automation in the form of driver assistance features. Here, the vehicle is able to provide assistance to the driver with either acceleration/deceleration or steering one at a time but is not able to do both simultaneously. Examples of Level 1 automation include adaptive cruise control, where the car can maintain a set speed and distance from the vehicle ahead, and lane-keeping assist, which helps the driver stay within their lane. Despite these automated features, the responsibility still lies with the human driver to monitor the driving environment and intervene when necessary.

1.2.3 Level 2: Partial Automation (Combined Automation)

Advancing further, Level 2 introduces combined automation. In this level, the vehicle is able to control both acceleration/deceleration and steering simultaneously under specific conditions. In this scenario, the driver must remain engaged and monitor the driving continuously. While automation alleviates some workload, the driver is responsible for taking over if the system encounters situations

it cannot handle. Level 2 autonomy showcases advanced capabilities, but it still requires that the human driver should be vigilant and ready to intervene when needed.

1.2.4 Level 3: Conditional Automation (Limited Self-Driving)

Level 3 marks a significant leap towards more advanced autonomous capabilities. At this level, the vehicle can manage most aspects of driving under certain conditions without requiring human intervention. The driver, however, can disengage from driving tasks and focus on non-driving activities, assuming the role of a passenger. However, Level 3 automation is conditional and typically confined to specific operational design domains (ODDs), limiting its capabilities to predefined scenarios. The challenge lies in regaining control when the system encounters situations outside its ODD or when requested by the driver or infrastructure.

1.2.5 Level 4: High Automation (Full Self-Driving in Limited Circumstances)

Moving closer to full autonomy, Level 4 represents high automation, where the vehicle can handle most driving tasks without human input or intervention. Unlike Level 3, Level 4 automation is not bound by specific operational domains; instead, it operates within predefined geo-fenced areas or controlled environments. Level 4 AVs offer a substantial increase in safety and convenience, making them ideal for specific use cases, such as autonomous shuttles in designated urban areas or self-driving trucks in controlled logistics yards.

1.2.6 Level 5: Full Automation (True Self-Driving)

At the pinnacle of the SAE Levels, Level 5 signifies full automation without any human involvement necessary for driving tasks. Level 5 AVs can operate under

any conditions and across all geographic areas, completely eliminating the need for a human driver. These vehicles are designed to provide passengers with a seamless and fully autonomous transportation experience. Level 5 autonomy represents the vision of fully realizing the potential of autonomous driving, revolutionizing urban mobility, enhancing road safety, and transforming the transportation landscape.

The real automated driving (AD) starts from level 3. At level 1 and 2, it is called Advanced driver-assistance system (ADAS) [10].

1.3 Modules of Autonomous Vehicles

There are five main modules in autonomous vehicles [11].

1.3.1 Sensing Module

The sensing module is a very important component of autonomous vehicles, endowing them with the capability of perceiving and interpreting their surroundings.

The most commonly used sensors used are:

1.3.1.1 Camera

The camera sensor is a crucial component of the sensing module in autonomous vehicles, mimicking the human eye's visual perception [12]. Mounted at strategic locations on the vehicle's exterior, these high-resolution cameras capture real-time images of the surrounding environment. The camera sensor plays a vital role in recognizing and understanding critical elements such as road signs, traffic signals, pedestrians, and other vehicles. Moreover, redundancy is often employed, with multiple cameras providing a comprehensive 360-degree view, ensuring robustness and reliability even in complex driving scenarios known as multi-camera setup [13]. For example, the Waymo Open Dataset car has 5 cameras, front, front-left,

front-right, side-left, side-right [14]. Stereo camera is also very commonly used so that distances can also be measured [15].

1.3.1.2 LiDAR

The use of LiDAR sensors in autonomous vehicles represents a groundbreaking advancement in self-driving technology. LiDAR, short for Light Detection and Ranging, utilizes laser pulses to create precise and detailed 3D maps of the vehicle's surroundings. Mounted on the vehicle's roof or integrated into its body, LiDAR sensors emit laser beams that bounce off surrounding objects and return to the sensor, providing accurate distance measurements. By scanning the environment at a rapid pace, LiDAR sensors generate real-time point clouds that offer a comprehensive view of nearby objects, road geometry, and potential obstacles. This rich data is then fused with information from other sensors, such as cameras and radars, to obtain an overall understanding of the AV's environment. LiDAR's ability to detect objects with high accuracy, even in low-light conditions, and its proficiency in distinguishing fine details make it a crucial sensor in ensuring the reliability and safety of autonomous vehicles. As the technology continues to evolve, LiDAR sensors are anticipated to play an increasingly vital role in shaping the future of self-driving transportation. For example, the Waymo Open Dataset car used five LiDARs, front, right, side-left, side-right, and top sensors [14].

1.3.1.3 RADAR

Radar technology has a fundamental role in the domain of autonomous driving, providing essential data for perceiving and navigating the vehicle's environment. Radar, which stands for Radio Detection and Ranging, uses radio waves to detect objects and measure their distance, speed, and direction. Mounted on various parts of the vehicle, such as the front, sides, and rear, radar sensors continu-

ously scan the surroundings, creating a real-time map of nearby objects and their movements. The ability to function effectively in various weather conditions, including rain, fog, and darkness, makes radar sensors reliable even in challenging environments [16]. In combination with other sensors like cameras and LiDAR, radar enhances the autonomous vehicle's perception capabilities, offering a redundant and comprehensive view of the surroundings. By accurately detecting and tracking moving objects, including vehicles, pedestrians, and cyclists, radar ensures safe and efficient decision-making, making it an integral part of the suite of sensors in autonomous driving systems.

1.3.1.4 Ultrasonic

Ultrasonic sensors are integral components of the sensing module in autonomous vehicles, providing essential close-range detection capabilities. The working principle of ultrasonic sensors is that they emit sound waves of high frequency and then measure the time difference between the emitted and received sound waves which come back to the sensor when they bounce back after hitting the nearby objects. Mounted on the vehicle's exterior, ultrasonic sensors are particularly useful for detecting obstacles in the immediate vicinity, such as parked cars, curbs, and pedestrians. They assist in parking maneuvers [17] and low-speed navigation, helping the autonomous vehicle avoid collisions and ensure safe and precise movements in tight spaces. While their range is limited compared to other sensors like radar and LiDAR, ultrasonic sensors excel in providing reliable and real-time feedback for low-speed operations, enhancing the AV's overall situational awareness and contributing to a seamless and secure autonomous driving experience.

1.3.1.5 IMU

Inertial Measurement Unit (IMU) sensors play a critical role in the accurate localization and motion tracking of autonomous vehicles. Comprising gyroscopes

and accelerometers, the IMU measures the vehicle's angular velocity and acceleration in three dimensions. By continuously monitoring these parameters, the IMU provides essential data about the vehicle's orientation, tilt, and movement. This information is crucial for determining the AV's position and trajectory, especially in situations where GPS signals may be unreliable, such as in tunnels or urban canyons. The integration of IMU sensors with other localization technologies, such as GPS and odometry, enhances the vehicle's overall position estimation accuracy and ensures a robust and reliable autonomous driving system [16]. The IMU sensors enable precise motion tracking, aiding in smooth navigation, and contributing to the safe and efficient operation of autonomous vehicles in diverse environments.

There are some other sensors that are also used in AVs but here are mentioned the most commonly used ones.

Please see the Table [1.1](#) for the summary.

Table 1.1: Comparison of the Sensors Used in AVs

Sensor	Advantages	Disadvantages
Camera	<ol style="list-style-type: none"> 1. Excellent perceivability 2. Availability of lateral velocity 3. Availability for distribution of color 	<ol style="list-style-type: none"> 1. Heavy computational load 2. Light interference 3. Heavily affected by weather 4. Unavailability of radial velocity
LiDAR	<ol style="list-style-type: none"> 1. 3D data 2. High precision 3. 360° horizontal field of view 	<ol style="list-style-type: none"> 1. Affected greatly by weather and background noise 2. Expensive
RADAR	<ol style="list-style-type: none"> 1. Longer distance of work 2. Availability of radial velocity 3. Weather robustness 	<ol style="list-style-type: none"> 1. Not applicable to static objects 2. Higher ratio of false alarms
Ultrasonic	<ol style="list-style-type: none"> 1. Low price 	<ol style="list-style-type: none"> 1. Very low resolution data 2. Not applicable for high speeds

1.3.2 Perception Module

The perception module's primary objective is to analyze and interpret the vast amounts of data collected from the various sensors. Advanced computer vision algorithms and machine learning techniques process this data, identifying objects, recognizing road signs, detecting lane markings, and categorizing obstacles. Through object detection and classification, the AV can distinguish between pedestrians, cyclists, vehicles, and static objects, enabling it to interact with its environment intelligently. Moreover, the perception module can accurately determine the distances and velocities of surrounding objects, critical for planning appropriate trajectories and avoiding collisions.

The perception module of autonomous vehicles is responsible for several crucial tasks that enable the vehicle to understand and interpret its surroundings accurately. These tasks include

1.3.2.1 Object Detection and Classification

The perception module must identify and classify various objects present in the vehicle's environment, such as pedestrians, cyclists, vehicles, and static obstacles. Accurate object detection is essential for the AV to interact safely with its surroundings and make informed decisions while navigating through traffic [18]. The objects to be detected first of all include road users which are mainly pedestrians and vehicles. But more than that, it should detect and interpret lane markings and road boundaries accurately. This information is crucial for the AV to maintain its position within the correct lane and follow the appropriate path while driving. Recognizing and interpreting traffic signs, including speed limits, stop signs, and traffic signals, is vital for the AV to adhere to traffic regulations and adjust its behavior accordingly.

1.3.2.2 Semantic Segmentation

Semantic segmentation is a computer vision task that involves dividing an image into multiple segments and assigning each pixel a semantic class label [18]. Unlike object detection, which recognizes objects as bounding boxes, semantic segmentation provides a fine-grained pixel-level understanding of the scene. This level of detail is invaluable in autonomous driving, as it allows AVs to differentiate between various objects, road markings, lane boundaries, and other critical elements in the environment, enhancing their perception and decision-making capabilities.

1.3.2.3 Environmental Mapping and Localization

The perception module creates detailed and dynamic high-definition maps of the vehicle's operational environment. These maps help the AV enhance its situational awareness and navigate accurately, especially in complex urban settings.

1.3.3 Prediction Module

As autonomous vehicles (AVs) become a reality on our roadways, the focus shifts beyond the technical challenges of navigation and perception to encompass understanding human behavior and intent. For AVs to safely and seamlessly coexist with human-driven vehicles, they must be capable of analyzing and predicting the behaviors of other road users, such as pedestrians, cyclists, and drivers. Behavior analysis and prediction play a critical role in ensuring smooth interactions, proactive decision-making, and, ultimately, enhancing the safety and acceptance of autonomous vehicles [19].

Human behavior on the road is diverse, complex, and often influenced by various factors, such as cultural norms, emotions, and the surrounding environment. Humans interpret and respond to subtle cues, such as eye contact, gestures, and body language, to predict the intentions of others. These nuanced interactions

are deeply ingrained in the driving experience and are vital for safe and efficient navigation in traffic.

The challenge for AVs lies in replicating this understanding of human behavior and predicting the intentions of other road users with a high degree of accuracy. This capability is essential for ensuring the AV can anticipate and react appropriately to dynamic traffic scenarios and avoid potential conflicts.

Machine learning and deep learning algorithms are often employed to recognize patterns in behavior and predict likely future actions of other road users. S. Mozaffari et. al., [20] have studied deep learning-based methods and techniques being used for this end. These algorithms use historical data and real-time observations to identify recurring behavior and make informed predictions. By constantly updating and refining their understanding of human behavior, AVs can navigate confidently and predictably in a variety of traffic situations.

The common tasks of behavior analysis and prediction modules are road scene understanding, driving activities classification, driving status classification, driver-road interaction classification, situational awareness classification, and trajectory prediction of road users other than the ego vehicle [21].

1.3.3.1 Trajectory prediction

Trajectory prediction, also known as trajectory forecasting, is also of immense importance. The idea is to observe the target vehicle, whose trajectory is to be predicted for some time, and then predict its future trajectory for some future time period. This is again mainly classified into pedestrian and vehicle trajectory prediction and usually, the models are different for both for the same task for trajectory prediction [19].

1.3.4 Planning Module

The planning module is a crucial component of autonomous vehicles, responsible for generating safe and efficient driving trajectories based on the information received from the perception module. Combining real-time data from various sensors and maps, the planning module processes this information to make intelligent decisions about the vehicle's actions and path. By considering factors like traffic conditions, road geometry, speed limits, and the behavior of other road users to plan smooth and collision-free routes. The planning module must balance the AV's mission to reach its destination efficiently while prioritizing safety and adhering to traffic rules. By synthesizing all these elements, the planning module ensures that autonomous vehicles can navigate complex urban environments, make informed decisions, and safely reach their destinations.

1.3.5 Control Module

The control module is the backbone of autonomous vehicles, responsible for translating the decisions made by the planning module into physical actions. It takes the planned trajectory and executes it by manipulating the vehicle's throttle, brakes, and steering system. The control module continuously adjusts these actions based on real-time feedback from various sensors to maintain stability, accuracy, and safety during driving. It ensures that the AV follows its intended path with precision, adheres to speed limits, and smoothly navigates around obstacles. The control module's efficiency is crucial in maintaining passenger comfort and confidence in the AV's capabilities. By seamlessly coordinating with other components, the control module plays a central role in realizing the promise of autonomous vehicles, offering a reliable and enjoyable driving experience while keeping safety at the forefront.

1.4 Deep Neural Networks

Deep Neural Networks (DNNs) have revolutionized the field of artificial intelligence, showcasing remarkable performance across diverse domains. These networks have demonstrated their ability to learn complex patterns and representations from data, making them the go-to choice for various tasks.

1.4.1 The Working of Deep Neural Networks

1.4.1.1 Architecture:

At the core of DNNs lies a layered architecture, with interconnected neurons in each layer. These neurons process and transmit information, forming the basis of computation in DNNs.

1.4.1.2 Feedforward Process:

The feedforward mechanism in DNNs involves passing input data through the network, layer by layer, to produce an output prediction. Weighted connections and activation functions shape the output and aid in modeling complex relationships in data.

1.4.1.3 Backpropagation:

The backpropagation algorithm is the foundation of learning in DNNs. This is an iterative process of updating weights based on the network's prediction error, enabling the network to improve its performance over time.

1.4.2 Types of Deep Neural Networks

1.4.2.1 Convolutional Neural Networks (CNNs)

CNNs have proven to be highly effective in computer vision tasks. These have a specialized architecture, featuring convolutional and pooling layers to capture spatial hierarchies in images.

1.4.2.2 Recurrent Neural Networks (RNNs)

RNNs are designed to handle sequential data, making them ideal for tasks with temporal dependencies. The concept of recurrent connections and the ability of RNNs is to maintain a memory of past inputs.

1.4.2.3 Autoencoders

Autoencoders are a class of DNNs used for unsupervised learning and data compression. These have encoder and decoder architectures, highlighting their applications in dimensionality reduction and data generation.

1.4.2.4 Generative Adversarial Networks (GANs)

GANs consist of a generator and discriminator network, trained in a competitive manner to generate realistic data. These have a role in generating synthetic images, enhancing data augmentation, and even creating deep fakes.

1.5 Recurrent Neural Networks

The fundamental architecture of an RNN consists of recurrent connections between neurons, forming a directed cycle within the network. These recurrent

connections enable the flow of information from one time step to the next, allowing RNNs to process sequential data. At each time step, the input is fed into the RNN, and the network computes an output and updates its hidden state based on the current input and the information retained from the previous time step. Since the input in this work is sequential and output is also sequential, this work is basically a sequential-to-sequential network and RNNs are used in both input and output side. At the input side it is called encoder as it encodes the sequential data into an encoded hidden state and the output side is called decoder as it again transforms the encoded hidden state into sequential output.

1.5.1 Forward Propagation

The forward propagation process in an RNN involves passing input data through the network, one time step at a time. At each time step, the network computes the output based on the current input and the hidden state from the previous time step. The computed output can be used for prediction or further processing, and the hidden state is updated to retain relevant information for the subsequent time step. This process is repeated until the entire sequence is processed.

1.5.2 Backpropagation Through Time (BPTT)

Training RNNs involves an extension of the backpropagation algorithm called Backpropagation Through Time (BPTT). BPTT computes the gradients of the loss function with respect to the network's parameters, enabling weight updates to improve the model's performance over time. However, training RNNs with standard BPTT can suffer from the vanishing gradient problem, which makes it challenging for the network to learn long-range dependencies in sequential data.

1.6 The Specific Task Targeted by This Thesis Work

This thesis work is in the domain of prediction module and specifically about the trajectory prediction of pedestrians.

1.7 Problem Formulation

Given a target (a road user), observe it for some time period, observation length, l_o . During each time step of l_o , generate a feature vector x , so that when $t \geq l_o - 1$ we have a sequence of these feature vectors in the form of a set $X_t =$

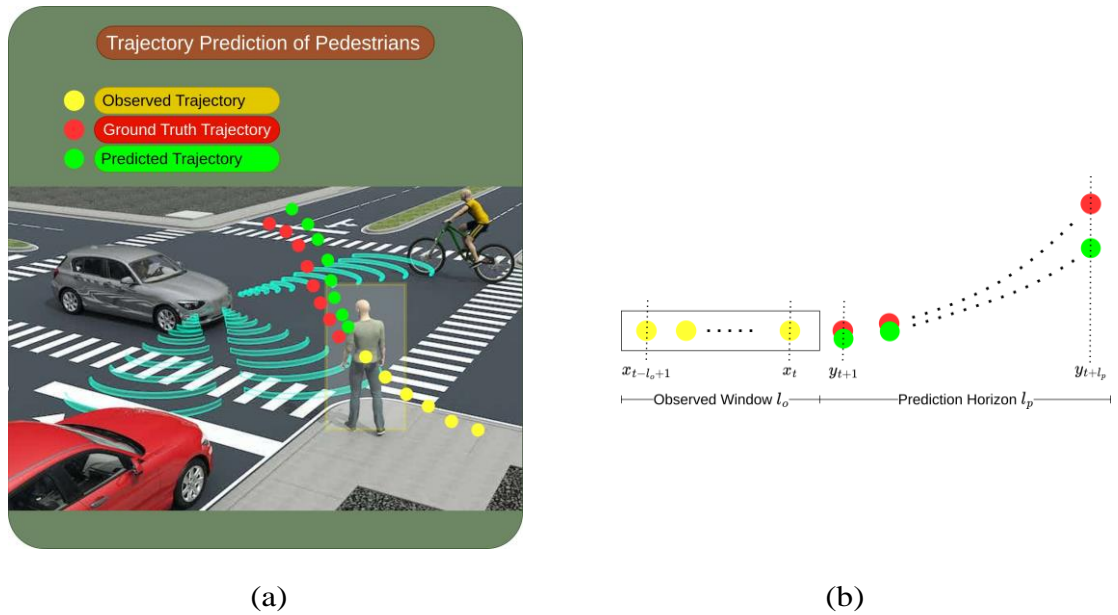


Figure 1.2: Illustration of the task of trajectory prediction of pedestrians for autonomous vehicles. **Yellow** represents observed trajectory, **Red** represents future ground truth trajectory, **Green** represents predicted trajectory. **(a)** Graphical illustration. **(b)** Symbolic illustration.

$\{x_{t-l_o+1}, x_{t-l_o+2}, \dots, x_t\}$. Our goal is to design a model f , which at time $t \geq l_o - 1$ will take this sequence X_t of past l_o feature vectors and generate future trajectory points in the form of vectors starting from time step $t + 1$ to $t + l_p$ where l_p denotes the time period of prediction horizon, so that we have a predicted trajectory $Y_t = f(X_t) = \{y_{t+1}, y_{t+2}, \dots, y_{t+l_p}\}$.

1.8 Research Objectives

1. Finding out the features which when added with spatial features improve the accuracy of the trajectory prediction of pedestrians for First Person View (FPV) ego-vehicle centric camera sensor Autonomous Vehicles.
2. Improving long term prediction.

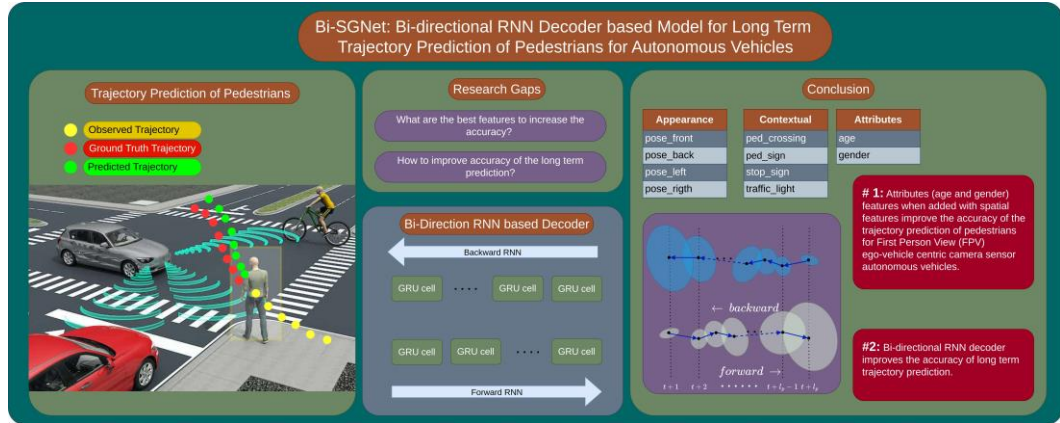


Figure 1.3: Graphical abstract, which depicts this work in a concise pictorial form. **1st Column (left)** illustrates the task of trajectory prediction of pedestrians for autonomous vehicles. **2nd Column (middle)** illustrates the research gaps found and targeted in this work and methodology. **3rd Column (right)** illustrates the conclusion.

Chapter 2: Literature Review

2.1 Classification of Methods of Trajectory Prediction

Researchers have pursued two main strategies: physics-based models, based on traditional motion theories and human behaviors, and learning-based models, exploiting machine learning algorithms and abundant data resources [20].

Stockem Novo et al. [10] have used the term data-driven models for learning-based models.

2.1.1 Physics-Based

Physics-based models use established principles of physics to predict trajectories. These models often use data such as velocity, acceleration, and road layout, and may incorporate behavioral models based on traffic rules and driver psychology.

Although physics-based models have many advantages like physics-based models are grounded in physical reality and human behavior, providing interpretable and explainable predictions [22]. Furthermore, they are computationally efficient as they do not require large training datasets or powerful hardware resources.

But, despite their advantages, physics-based models often fall short in predicting complex or erratic behavior due to their dependence on predefined rules and assumptions about the environment. The deterministic nature of physics-based models limits their ability to represent the inherent uncertainty in human behavior and road conditions [19].

2.1.2 Learning Based

In contrast, learning-based models leverage machine learning algorithms to learn from historical data and make predictions. Deep learning techniques like Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks are popular in this field [23].

Learning-based models have the ability to learn complex patterns and adapt to changing environments, resulting in superior performance in complex and dynamic scenarios [24]. They are capable of estimating uncertainties in trajectory prediction through probabilistic frameworks, providing a measure of confidence in their predictions [25]. That's why we have chosen a learning-based model for our work.

2.2 Unimodal vs Multimodal Trajectory Prediction

2.2.1 Unimodal Trajectory Prediction

Unimodal trajectory prediction posits a single, most likely trajectory for each actor. This approach assumes deterministic future behavior and is typically employed in model-based methods such as Kalman Filters and its variants and also in learning-based models like [26], [27]. While these methods are computationally efficient, they often lack the ability to predict multiple potential futures, which may reduce their robustness in complex and dynamic traffic scenarios.

2.2.2 Multimodal Trajectory Prediction

Contrary to unimodal prediction, multimodal trajectory prediction aims to model a range of potential future paths for each actor, accounting for the inherent uncer-

tainty in predicting human behavior [28]. This is usually accomplished through probabilistic methods, such as Gaussian Mixture Models (GMMs) [29], [30], [31] and Generative Adversarial Networks (GANs). Despite being more computationally intensive, these methods provide a more robust understanding of potential future scenarios. This work focuses on multimodal trajectory prediction which are stochastically generated.

2.3 Multi-Agent vs. Single-Agent Trajectory Prediction

2.3.1 Multi-Agent Trajectory Prediction

This approach recognizes the interdependencies among various actors within a traffic environment and considers their mutual interactions to make more accurate predictions. Techniques such as Social LSTM [23] and [26] and Graph Neural Networks (GNNs) [32] have been developed to model these complex inter-agent relationships. For instance, Social LSTM applies recurrent neural networks to capture the dynamic social interplay between pedestrians, cyclists, and vehicles, thereby improving prediction accuracy. Similarly, GNNs model the interactions among traffic participants as a graph, allowing for a more intuitive representation of their mutual influences [33]. These multiagent methods offer a more holistic understanding of road users' behaviors and have the potential to significantly enhance the safety and efficiency of AV operations.

2.3.2 Single-Agent Trajectory Prediction

Single-agent prediction focuses on forecasting the trajectory of individual road users independently. Although they may include interactions with other agents, environment, and context, the main differentiator is that the model creates a

trajectory of only one agent at a time. This work is about single-agent trajectory prediction.

2.4 Context Awareness

In the trajectory prediction for autonomous vehicles (AVs), context awareness plays a significant role in achieving reliable and safe navigational decisions. Context awareness refers to the process of considering the environmental and situational factors surrounding an agent, such as road infrastructure, traffic rules, and other road users' behavior, when predicting its future trajectory [34]. Machine learning techniques, particularly deep learning, have been instrumental in integrating such contextual information into trajectory prediction models. Convolutional Neural Networks (CNNs), for instance, can encode high-dimensional sensor data to capture spatial context [35]. Similarly, methods like Social LSTM incorporate social context, i.e., the interactions among multiple agents, to enhance prediction accuracy [23]. Moreover, scene-specific methods that employ high-definition maps or semantic scene understanding can provide rich context information, leading to more informed predictions [36].

2.5 Goal Driven/Conditioned

Goal-driven trajectory prediction is an emerging approach in autonomous vehicle (AV) technology that aims to predict future trajectories of road users by identifying their objectives or 'goals'. This approach acknowledges that an agent's future path is largely dictated by their intended destination or objective [37]. There are several types of goal-driven trajectory prediction models.

2.5.1 End Goal

Firstly, end-goal prediction techniques infer the final destination of an agent based on their current and past positions, and subsequently predict the path leading to this end-goal [36]. This type of prediction is particularly useful in scenarios where the agent's goal can be reasonably inferred from the onset, such as highway driving.

2.5.2 Grid Based

Secondly, grid-based methods divide the environment into a grid and assign each cell a likelihood of being the agent's goal [38]. The trajectory is then predicted based on the most likely goal cells. This method is particularly useful in complex environments where the end goal may not be immediately apparent, such as urban intersections or parking lots.

2.5.3 Stepwise Goals

Lastly, stepwise goal prediction models generate a series of sub-goals that guide the agent's trajectory [39]. This approach is beneficial in dynamic environments where immediate future goals might be more discernible than distant end goals. The baseline method [2] for this work is also using this stepwise goals approach.

Table 2.1: Literature Review of Methods

Classification	Classes	Published Works
Methods	Physics-Based	[22]
	Learning-Based	[25], [23], [24]
Unimodal vs.	Unimodal	[26], [27]
Multimodal	Multimodal	[28], [29], [30], [31]
Single- vs.	Single-Agent	[2]
Multi-Agent	Multi-Agent	[23], [26], [32], [33]
Context Awareness	-	[34], [35], [23], [36]
	End Goal	[36]
Goal Conditioned	Grid Based	[38]
	Stepwise Goals	[39], [2]

2.6 Datasets

2.6.1 Sensors-based Classification

Some datasets have a full suite of autonomous vehicle sensors which include LiDARs, multiple cameras, RADAR, GPS, Gyroscope/IMU, etc. Such datasets are nuScenes [40], Waymo Open Dataset [14], Agroverse [41], Lyft Level5 [42], Shifts Vehicle Motion Prediction by Yandex [43] etc.

While other datasets include only some or one sensor instead of the full suite. Among those which include only the camera sensor like *JAAD* [] and *PIE* [] dataset.

2.6.2 Type of Target-based Classification

The datasets which are specifically designed for the task of trajectory prediction can be classified into their targets.

Pedestrians like [11], *JAAD* [4], *PIE* [3], Stanford Drone Dataset [44], [45], [46]. Ranga et al. [11] use the term vulnerable road users which is a broader term than pedestrians and includes cyclists, scooters etc also.

Vehicles like nuScenes [40], Waymo Open Dataset [14], Agroverse [41], Lyft Level5 [42], Shifts Vehicle Motion Prediction by Yandex [43].

Both like nuScenes [40].

2.6.3 Point of View-based Classification

2.6.3.1 *First Person View (FPV)*

The datasets in which the camera is ego-vehicle-centric. For example in *JAAD* [4] and *PIE* [3] datasets.

2.6.3.2 *Bird's Eye View (BEV)*

The datasets which are top view. These are either captured from aerial footage like Stanford Drone Dataset [44], [45], [46]. In the nuScenes dataset, [40], they have provided BEV of LiDAR as an additional option. Similarly, in the KITTI dataset [47], they have provided BEV detection.

2.6.4 Driving Environment-based Classification

Driving environment means the outside external environment and conditions where the autonomous vehicle is intended to operate. These are classified as

2.6.4.1 *Highways*

There are some datasets that are just for highways like NGSIM [48]. The Highway scenario has its own challenges. Mostly, the traffic is very straight forward but because of high speeds, any miscalculation can lead to disastrous consequences.

2.6.4.2 *Rural or Urban Driving Environment*

In rural areas, factors such as limited road infrastructure, unpredictable wildlife, and adverse weather conditions necessitate robust sensing, perception, and decision-making capabilities. On the other hand, urban areas introduce complex scenarios involving dense traffic, diverse road users, and intricate road markings, demanding precise navigation and real-time interaction with pedestrians and cyclists. Most of the datasets for autonomous vehicles are for the urban environment. DATS dataset [49] is a dataset recorded for the Indian region for both rural and urban driving environments.

2.6.4.3 *Time of the Day*

Time of the day is also very important mainly classified as:

- Morning time.
- Day time.
- Evening time.
- Night time.

- Etc.

Most of the datasets try to have scenarios for all above mentioned time of the day.

2.6.4.4 *Weather*

Weather is also a very important factor. It has been classified as:

- Sunny.
- Rainy.
- Snowy.
- Foggy.
- Etc.

Most of the datasets try to include all weather conditions. But, recently, a dataset Canadian Adverse Driving Conditions Dataset (CADDC) [50] has been proposed which specifically targets wintry conditions which always include snowfall.

Table 2.2: Literature Review of Datasets

Classification	Classes	Published Works
Sensors	Full Suite	nuScenes [40], Waymo Open Dataset [14], Agrove [41], Lyft Level5 [42], Shifts Vehicle Motion Prediction by Yandex [43]
	Camera Only	JAAD [4], PIE [3]
Target Road User	Pedestrians	JAAD [4], PIE [3], Stanford Drone Dataset [44], ETH [45], UCY [46]
	Vehicles	nuScenes [40], Waymo Open Dataset [14], Agrove [41], Lyft Level5 [42], Shifts Vehicle Motion Prediction by Yandex [43]
	Both	nuScenes [40]
Point of View	FPV	JAAD [4], PIE []
	BEV	Stanford Drone Dataset [44], ETH [45], UCY [46], KITTI [47]
Driving Environment	Highways	NGSIM [48]
	Rural	DATS [49]
	Wintry	Canadian Adverse Driving Conditions Dataset
	Weather	(CADC) [50]

Chapter 3: Methodology

3.1 Datasets Selection

Two datasets *JAAD* [4] and *PIE* [3] have been selected for this work. Their classification as per the Section 2.6 classification is given in the following Table 3.1

Table 3.1: Selected Datasets

↓Classes\Datasets→	<i>JAAD</i>	<i>PIE</i>
Target	Pedestrians	Pedestrians
Sensors	1 x Front Camera	1 x Front Camera
Point of View	FPV	FPV
Driving Environment	Urban	Urban

3.1.1 JAAD Dataset

The properties of the camera used are shown in the Table 3.2. The camera was installed below the rear view mirror inside the car.

Table 3.2: *JAAD* Dataset Camera Sensor Details

Camera Sensor	Type	Resolution	FOV	No. of Clips
Garmin GDR-35	Monocular	1920 x 1080	110°	276
GoPro Hero+	Monocular	1920 x 1080	170°	60
Highscreen BB Connect	Monocular	1920 x 1080	100°	10

Some more information is given in the following Table 3.3.

Table 3.3: *JAAD* Dataset Details

Size	4.3 GB
No of Video Clips	346
Locations	North America (60 video clips) and Eastern Europe (86 video clips)
Frames per second (FPS)	30
No of frames	Approximately 82k
No of unique pedestrians	2.2k
No of Bounding boxes	337k

3.1.2 PIE

The camera sensor is installed inside the car below the rear view mirror. Its properties are given in the Table 3.4.

Table 3.4: *PIE* Dataset Camera Sensor Details

Camera Sensor	Type	Resolution	FOV
Waylens Horizon	Monocular	1920 x 1080	157°

It has 6 sets of video clips as mentioned in the following Table 3.5

Table 3.5: *PIE* Dataset [3] Sets of Video Clips

Sets	No. of Clips
Set1	4
Set2	3
Set3	19
Set4	16
Set5	2
Set6	9
Total	53

Some more information is given in the following Table 3.6

Table 3.6: *PIE* Dataset [3] Details

Size	70.4 GB
No of Video Clips	53
Locations	Toronto, Canada
Hours of Driving	Over 6 hours
Frames per second (FPS)	30
No of frames	91k
No of unique pedestrians	1.2k
No of Bounding boxes	740k

3.2 Baseline Method

This work refers to [2] for the baseline method. It has two types of methods:

1. Deterministic, which is unimodal prediction.
2. Stochastic, which is multimodal trajectory prediction.

In this work, the stochastic method is used as the baseline method. The model SGNet-ED in [2] is the best in the state-of-the-art models for the stochastic methods for the selected datasets. Table 3.8 shows the comparison of the baseline method on the selected datasets.

Table 3.7: Classification of the Baseline Method SGNet-ED [2]

Target	Pedestrians
Point of View	FPV
Unimodal vs. Multimodal	Multimodal
Single- vs. Multi-Agent	Single-Agent
Context Awareness	No
Goal Conditioned	Stepwise goals

Table 3.8: Comparison of the Baseline Method SGNet-ED [2] with other State-of-the-Art Methods on *JAAD* [4] and *PIE* [3] Datasets (\downarrow lower the better)

Method (Best of 20)	<i>JAAD</i>					<i>PIE</i>				
	<i>MSE</i> \downarrow			<i>C_{MSE}</i> \downarrow	<i>CF_{MSE}</i> \downarrow	<i>MSE</i> \downarrow			<i>C_{MSE}</i> \downarrow	<i>CF_{MSE}</i> \downarrow
	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s
Linear [3]	233	857	2303	1565	6111	123	477	1365	950	3983
LSTM [3]	289	569	1558	1473	5766	172	330	911	837	3352
PIE_{traj} [3]	110	399	1280	1183	4780	58	200	636	596	2477
PIE_{full} [3]	-	-	-	-	-	-	-	556	520	2162
BiTraP-D	93	378	1206	1105	4565	41	161	511	481	1949
BiTrap-GMM	153	250	585	501	998	38	90	209	171	368
BiTrap-NP	38	94	222	177	565	23	48	102	81	261
SGNet-ED [2]	37	86	197	146	443	16	39	88	66	206

Note: The methods in 2nd row are deterministic and in 3rd row are stochastic methods.

3.3 Model Architecture

We represent a fully connected layer (FCL) as follows:

$$\mathbf{Y} = FCL(\mathbf{X})_{\mathbf{W},\mathbf{b}} \quad (1)$$

read as "Y is equal to FCL of X with weights W and bias b" which is defined as:

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^T + \mathbf{b} \quad (2)$$

where $\mathbf{X} \in \mathbb{R}^{D \times l_{in}}$, $\mathbf{W} \in \mathbb{R}^{l_{out} \times l_{in}}$, $\mathbf{b} \in \mathbb{R}^{l_{out}}$, $\mathbf{Y} \in \mathbb{R}^{D \times l_{out}}$ and D is any dimensionality including none, l_{in} is the number of dimensions of input vectors contained in the input tensor \mathbf{X} , l_{out} is the number of dimensions of output vectors contained in the output tensor \mathbf{Y} . Since the dimensions of the bias \mathbf{b} , have no degree of freedom, thus its dimensions will not explicitly mentioned.

The training is done in batches, but the methodology is explained for the batch size of 1, which means that the batch size can just be ignored. But, for the batch size greater than 1, the first dimension of every dimension will be the batch size itself and all equations still hold. Also, whenever a tensor is reshaped, its notation will not be changed.

3.3.1 Data Generation

Data is generated from the data set in the form of corresponding input samples and output labels. One input sample is a sequence of feature vectors which represents the trajectory of a pedestrian during the observation time period generated

from the consecutive frames of the observation time period. The corresponding output label of that input sample is a set of sequences where a sequence is representing the trajectory of the future prediction horizon of the pedestrian generated from the consecutive frames of the prediction time period. So, for every time step of the input sequence, there is a corresponding output sequence spanning the complete prediction horizon time steps.

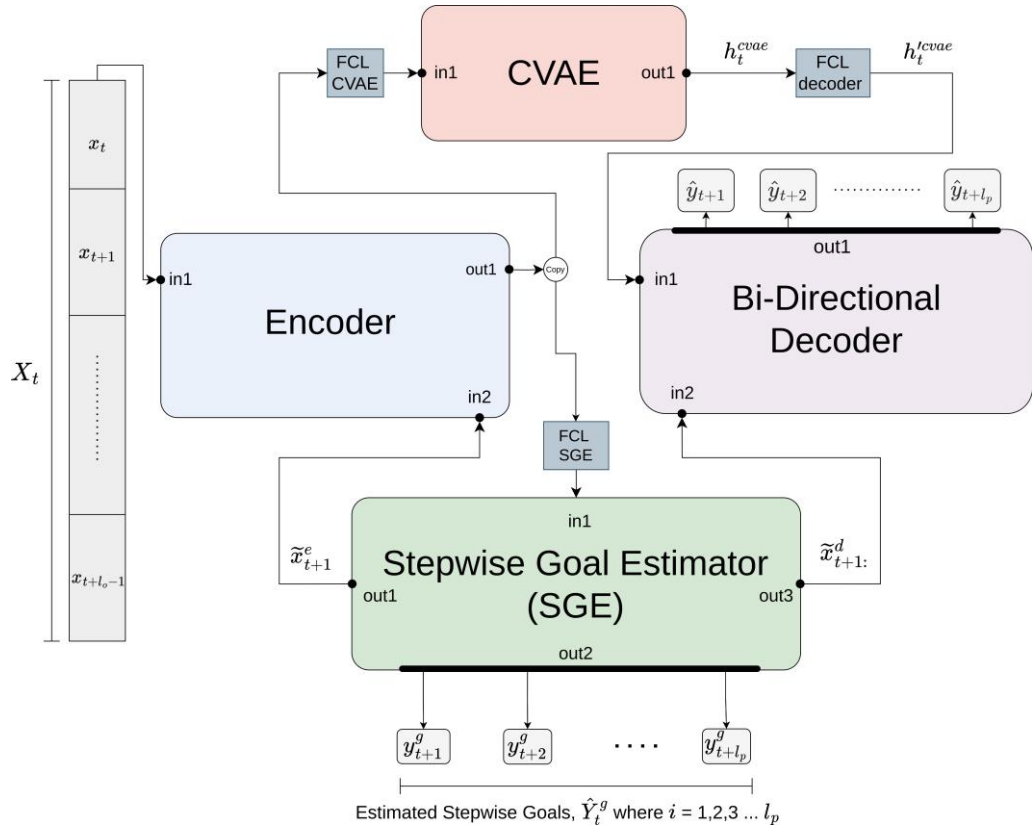


Figure 3.1: Block diagram of the model architecture. All blocks are explained in detail in the later sections. All inputs and outputs are labeled and can be matched. (1) Encoder, detailed diagram in Figure 3.2 (2) Stepwise goal estimator (SGE), detailed diagram in Figure 3.3 (3) Conditional Variational Autoencoder (CVAE), detailed diagram in Figure 3.4 (4) Bi-directional Decoder, detailed diagram in Figure 3.5

$$\forall \mathbf{x}_t \in \mathbf{X}_t, \text{ there exists } \mathbf{Y}_t = \{\mathbf{y}_{t+1}, \mathbf{y}_{t+2}, \dots, \mathbf{y}_{t+l_p}\} \quad (3)$$

where $\mathbf{x} \in \mathbb{R}^{l_x}$ is a feature vector of time step, t , of an input sequence $\mathbf{X}_t \in \mathbb{R}^{l_o \times l_x}$, \mathbf{Y} is the set of future trajectory's points starting from the next time step, $t + 1$, till the $(t + l_p)^{th}$ time step, a future trajectory's point is represented by a vector \mathbf{y}_{t+i} where $i \in [1, l_p]$, l_o is the length of the observation time period and l_p is the length of the prediction horizon.

One sample of the input sequences $\mathbf{X}_t \in \mathbb{R}^{l_o \times l_x}$ where l_o is observation length, l_x is a number of input feature vector dimensions.

3.3.2 Encoder

Consider an input feature vector at time step t , $\mathbf{x}_t \in \mathbb{R}^{l_x}$. It will be first embedded into the embedding layer to represent it as a latent representation.

$$\mathbf{x}_t^f = \text{ReLU}(FCL_{emb}(\mathbf{x}_t) \mathbf{W}_{emb, b_{emb}}) \quad (4)$$

where $\mathbf{W}_{emb} \in \mathbb{R}^{l_{he} \times l_x}$, $\mathbf{x}_t^f \in \mathbb{R}^{l_{he}}$, l_{he} is hidden size of encoder's GRU cell.

This is then concatenated with the goal estimated by the stepwise goal estimator (SGE) from the previous time step $t - 1$, for this time step, t .

$$\mathbf{x}_t^e = \mathbf{x}_t^e \oplus \mathbf{x}_t^f \quad (5)$$

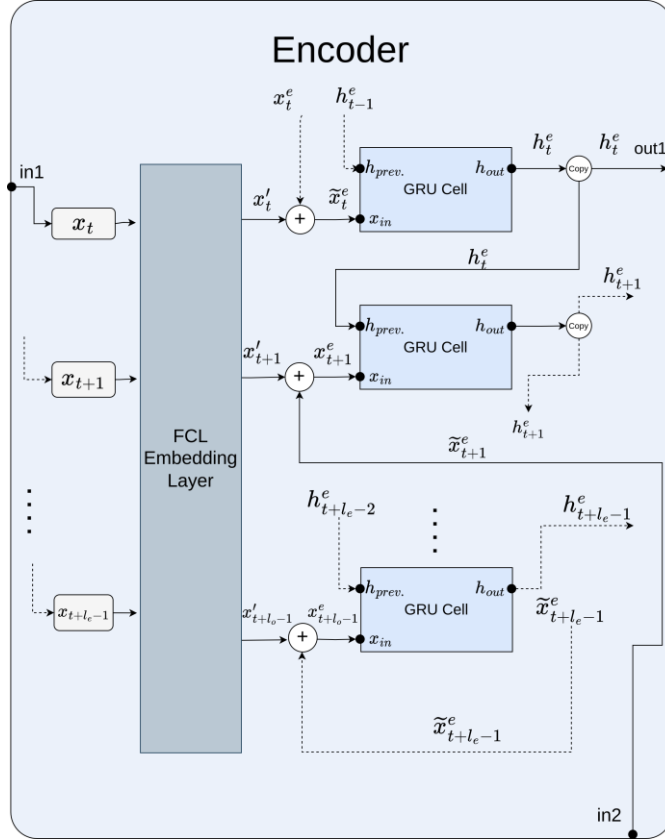


Figure 3.2: Detailed diagram of the encoder module. All inputs and outputs match with the block diagram shown in Figure 3.1

where $\tilde{\mathbf{x}}_t^e \in \mathbb{R}^{l_{SGE}}$ is the goal estimated by the stepwise goal estimator (*SGE*) from the previous time step $t - 1$, for this time step, t , l_{SGE} is the length of the hidden size of *SGE*. For the first time step, it is initialized with zeros.

The concatenated result $\mathbf{x}_t^e \in \mathbb{R}^{(l_{he}+l_{SGE})}$ is then fed into the encoder's GRU cell.

$$\mathbf{h}_t^e = GRU^{enc}(\mathbf{x}_t^e, \mathbf{h}_{t-1}^e) \quad (6)$$

where $\mathbf{h}_{t-1}^e \in \mathbb{R}^{l_{he}}$, GRU^{enc} is a GRU cell with hidden size l_{he} and input size $l_{SGE} + l_{he}$. The output hidden state of the GRU^{enc} is \mathbf{h}_t^e with the same dimension

as h_{t-1}^e .

At this point, the h_t^e will follow three different paths.

3.3.3 Stepwise Goal Estimator (SGE)

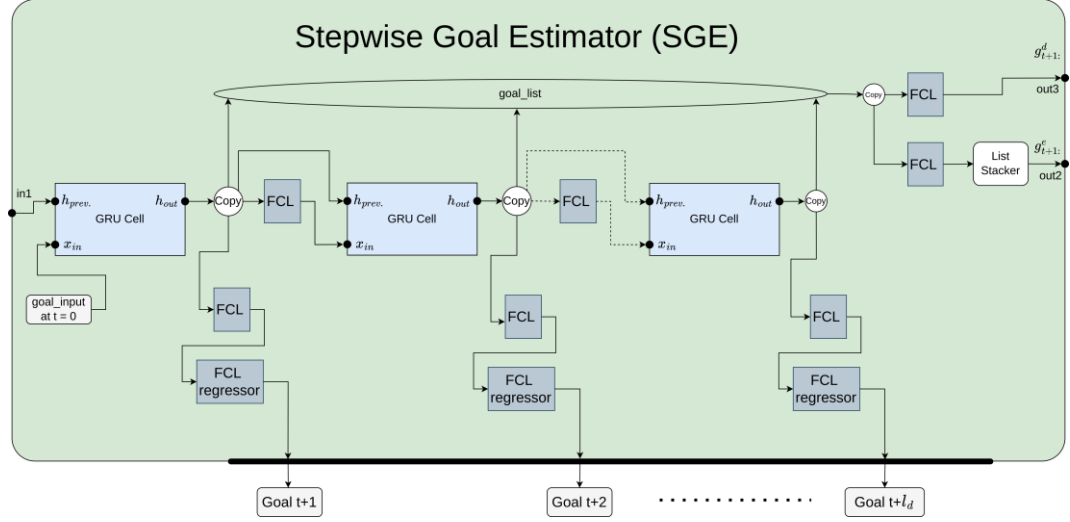


Figure 3.3: Detailed diagram of the stepwise goal estimator (SGE) module. All inputs and outputs match with the block diagram shown in Figure 3.1

First path, h_t^e is fed into SGE , through an FCL .

$$\mathbf{h}_t^g = \text{ReLU}(FCL_{SGE}(\mathbf{h}_t^e) \mathbf{w}_{SGE}, \mathbf{b}_{SGE}) \quad (7)$$

where $\mathbf{w}_{SGE} \in \mathbb{R}^{l_{SGE} \times l_{h_e}}$, $\mathbf{h}_t^g \in \mathbb{R}^{l_{SGE}}$

The SGE module takes one input and gives three outputs. The SGE module as used in SGNet-ED in [2] has been modified with the modification that all the attention mechanism based goal aggregator part has been put inside the SGE module. So, the outputs of the SGE module are the same as SGNet-ED's SGE

module except the attention mechanism based goal aggregator part is already done inside the *SGE* module instead of doing outside it.

h_t^g from equation 7 is the input of the *SGE* module.

$$\mathbf{x}_{t+1}^e, \hat{\mathbf{Y}}_t^g, \mathbf{x}_{t+1:p}^d = f_{SGE} (h_t^g) \quad (8)$$

The outputs are:

1) goal for encoder in latent vectors form only for next time step, $\mathbf{x}_{t+1}^e \in \mathbb{R}^{l_{SGE}}$. This will be fed back into the *Encoder* module as it's 2nd input to play its part in the input of next time step $t + 1$'s encoder's GRU cell. But, this specific processing step will not occur now, it will be the final step after all the processing of the decoder is finished.

2) set of stepwise estimated goals for future trajectory $\hat{\mathbf{Y}}_t^g$, for each time step from $t + 1$ to $t + l_p$ in dimensions same as that of predicted future trajectory, In generalized form, the set of stepwise future estimated goals for each time step from $t + 1$ to $t + l_p$ is:

$$\hat{\mathbf{Y}}_t^g = \{ \mathbf{y}_{t+1}^g, \mathbf{y}_{t+2}^g, \dots, \mathbf{y}_{t+l_p}^g \} \quad (9)$$

where $\tilde{\mathbf{y}}_{t+i}^g \in \mathbb{R}^{l_y}, i \in [1, l_p]$

This is saved for each time step and will be used to calculate *Loss* function in the end of one sequence of observation time period.

3) set of estimated goal for decoder in latent vectors form from next time step $t + 1$ to $t + l$, $\mathbf{x}_{t+1:p}^d$

In generalized form, the set of future estimated goals for each time step from $t + 1$ to $t + l_p$ for decoder is

$$\mathbf{x}_{t+1:p}^d = \{ \tilde{\mathbf{x}}_{t+1}^d, \tilde{\mathbf{x}}_{t+2}^d, \dots, \tilde{\mathbf{x}}_{t+l_p}^d \} \quad (10)$$

where $\tilde{\mathbf{x}}_{t+i}^d \in \mathbb{R}^{h_{SGE}}$, $i \in [1, l_p]$

For further processing, $\tilde{\mathbf{x}}_{t+1}^d$ is converted into a tensor $\in \mathbb{R}^{p \times h_{SGE}}$ which will be then fed into the decoder as its 2^{nd} input.

3.3.4 Conditional Variational Autoencoder (CVAE)

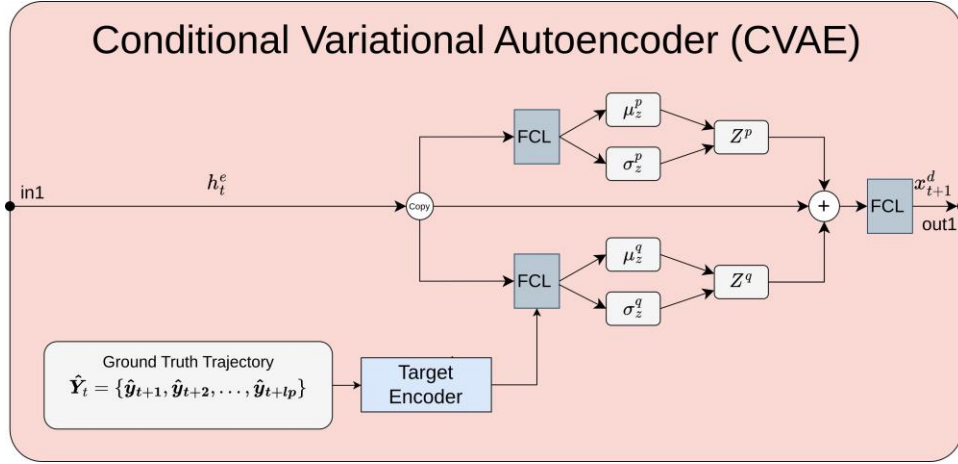


Figure 3.4: Detailed diagram of the conditional variational autoencoder (CVAE) module. All inputs and outputs match with the block diagram shown in Figure 3.1

We used exactly the same architecture as the baseline method [2] for *CV AE* module. The purpose of *CV AE* module is to learn distribution of the input data, conditioned on the distribution of corresponding ground truth trajectories and generate K new samples from that distribution, stochastically, which will be used to predict new trajectories. This is a way to stochastically add the multimodal trajectory prediction capability to the model. So, the model will predict K trajectories for each input trajectory instead of one and *CV AE* module is adding this stochastic multimodal trajectory prediction capability to the model.

Second path of h_t^e , as mentioned in Section 3.3.2, is fed into *CVAE* module through an FCL, which takes one input and returns three outputs.

$$h_t^{cvae}, KLD, \mathbf{p} = f_{CVAE}(ReLU(FCL_{CVAE}(h_t^e)_{W_{CVAE}, b_{CVAE}})) \quad (11)$$

where $h_t^{cvae} \in \mathbb{R}^{K \times (l_{he} + l_{LD})}$ is the output latent state vector of the *CVAE*, l_{LD} is the dimension of the latent space of *CVAE*, KLD is the KL divergence between the prior and recognition networks of *CVAE* as explained in [2], \mathbf{p} is probability vector containing the probability of each one of the K generated latent state vectors which are generated by *CVAE* module by taking h_t^e as input.

3.3.5 Bi-directional Decoder

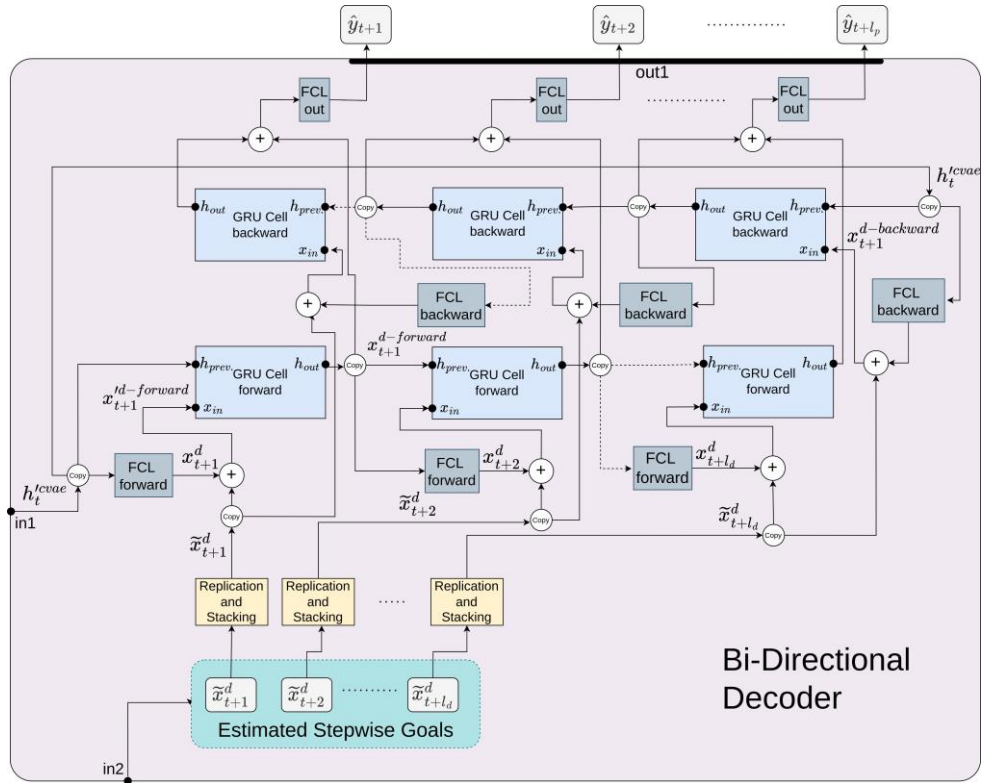


Figure 3.5: Detailed diagram of the bi-directional decoder module. All inputs and outputs match with the block diagram shown in Figure 3.1

The feature selection part of this work is done with the same *Decoder* as used in the baseline method [2]. But, then a new architecture of the decoder has been proposed, a bi-directional decoder. Here, the proposed architecture has been explained.

There are two inputs to decoder. The first one is the \mathbf{h}_t^{cvae} which is the output of *CVAE* module. \mathbf{h}_t^{cvae} is then passed through an FCL before being fed into the decoder.

$$\mathbf{h}_t^{cvae} = \text{ReLU}(FCL_{decoder}(\mathbf{h}_t^{cvae})_{\mathbf{W}_{decoder}, \mathbf{b}_{decoder}})) \quad (12)$$

where $\mathbf{W}_{decoder} \in \mathbb{R}^{l_{hd} \times (l_{he} + l_{LD})}$, $\mathbf{h}_t^{cvae} \in \mathbb{R}^{K \times l_{hd}}$

\mathbf{h}_t^{cvae} it is then fed into an FCL inside decoder.

$$\mathbf{x}_{t+1}^d = \text{ReLU}(FCL_{forward}(\mathbf{h}_t^{cvae})_{\mathbf{W}_{forward}, \mathbf{b}_{forward}})) \quad (13)$$

where $\mathbf{W}_{forward} \in \mathbb{R}^{l_{hd} \times l_{hd}}$, $\mathbf{x}_{t+1}^d \in \mathbb{R}^{K \times l_{hd}}$

The second input to the decoder is the goal vector $\bar{\mathbf{x}}_{t+1}^d$: which is the third output of the *SGE* module as explained in Section 3.3.3.

Now, \mathbf{x}^d is looped over $i \in [1, l_p]$. So, consider the case when $i = 1$. Here we have $\bar{\mathbf{x}}_{t+1}^d \in \mathbb{R}^{l_{SGE}}$. Since the first input to the decoder \mathbf{h}_t^{cvae} has K stochastic generations, \mathbf{x}_{t+1}^d is also required to have K vectors. But here it is not done by generation, it is done by replication. So, $\bar{\mathbf{x}}_{t+1}^d$ is replicated K times and stacked vertically. So, now we have $\bar{\mathbf{x}}_{t+1}^d \in \mathbb{R}^{K \times l_{SGE}}$. It is then concatenated with \mathbf{x}_{t+1}^d from equation 13.

$$\mathbf{x}_{t+1}^{d-forward} = \mathbf{x}_{t+1}^d \oplus \bar{\mathbf{x}}_{t+1}^d \quad (14)$$

where $\mathbf{x}_{t+1}^{d-forward} \in \mathbb{R}^{K \times (l_{SGE} + l_{hd})}$.

Our proposed bi-directional decoder as the decoder module which have forward and backward RNNs. So, $\mathbf{x}_{t+1}^{d-forward}$ is fed to the forward RNN which is a GRU cell as initial input and \mathbf{h}_t^{cvae} from 12 will be the initial hidden state.

$$\mathbf{h}_{t+1}^{d-forward} = GRU_{forward}^{enc}(\mathbf{x}_{t+1}^{d-forward}, \mathbf{h}_t^{cvae}) \quad (15)$$

where $\mathbf{h}_{t+1}^{d-forward} \in \mathbb{R}^{K \times l_{hd}}$.

$\mathbf{h}_{t+1}^{d-forward}$ will be then fed into the next time step of $GRU_{forward}^{enc}$ cell and so on up to the last time step $t + l_p$ and output hidden state will be saved at every time step to generate final predicted trajectories as the output of the model.

The backward RNN is also a GRU cell which starts from time step $t + l_p$. At this time step, the first input to decoder \mathbf{h}_t^{cvae} from 12 is first passed through an FCL.

$$\mathbf{x}_{t+l_p}^{d-backward} = ReLU(FCL_{backward}(\mathbf{h}_t^{cvae}) \mathbf{W}_{backward}^b) \quad (16)$$

where $\mathbf{W}_{backward} \in \mathbb{R}^{l_{hd} \times l_{hd}}$, $\mathbf{x}_{t+l_p}^{d-backward} \in \mathbb{R}^{K \times l_{hd}}$

$\mathbf{x}_{t+l_p}^{d-backward}$ is then concatenated with the last time step's $\mathbf{x}_{t+l_p}^d$ which is basically the last time step of the second input to the decoder \mathbf{x}_{t+1}^d from 3.3.3 on which

we have already completed the forward RNN sequence and now starting from its last time step $t + l_p$ for backward RNN.

$$\mathbf{x}_{t+l_p}^{d-backward} = \mathbf{x}_{t+l_p}^d \oplus \mathbf{x}_{t+l_p}^{d-backward} \quad (17)$$

where $\mathbf{x}_{t+l_p}^{d-backward} \in \mathbb{R}^{K \times (l_{SGE} + l_{hd})}$

$\mathbf{x}_{t+l_p}^{d-backward} \in \mathbb{R}^{K \times (l_{SGE} + l_{hd})}$ is then fed to the backward $GRU_{backward}^{dec}$ cell as initial input and \mathbf{h}_t^{cvae} from 12 will be the initial hidden state.

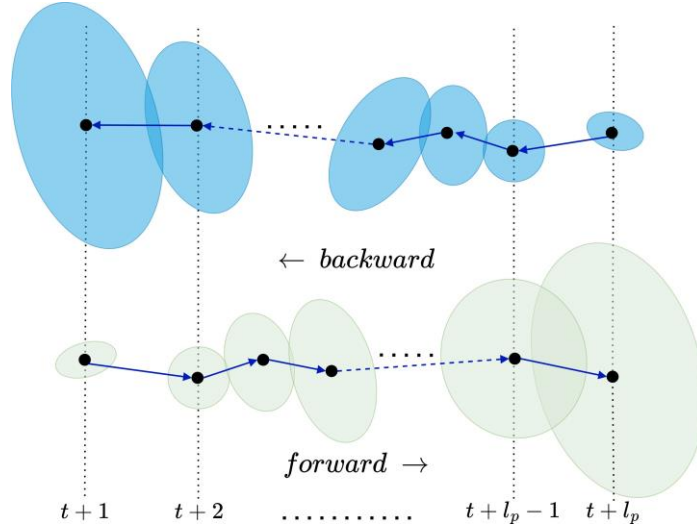


Figure 3.6: Symbolic working of bi-directional RNN. The ellipses represent the accumulated error. Forward pass is shown in green, and are increasing from time step $t + 1$ to $t + l_p$. Backward pass is shown in blue, and it is greater for time step $t + 1$ to $t + l_p$.

$$\mathbf{h}_{t+l_p}^{d-backward} = GRU_{backward}^{dec}(\mathbf{h}_t^{kvae}, \mathbf{x}_{t+l_p}^{d-backward}) \quad (18)$$

$\mathbf{h}_{t+l_p}^{d-backward}$ is then concatenated with the output hidden state of the forward $GRU_{forward}^{enc}$ cell at time step $t + l_p$ which has been already calculated in the forward pass of forward $GRU_{forward}^{enc}$ cell.

$$\tilde{\mathbf{y}}_{t+l_p}^f = \mathbf{h}_{t+l_p}^{d-forward} \oplus \mathbf{h}_{t+l_p}^{d-backward} \quad (19)$$

where $\tilde{\mathbf{y}}_{t+l_p}^f \in \mathbb{R}^{K \times (l_{h_d} + l_{h_d})}$

$\tilde{\mathbf{y}}_{t+l_p}^f$ is then passed through an FCL to generate the final time step vector of the predicted trajectory.

$$\hat{\mathbf{y}}_{t+l_p} = \text{ReLU}(\text{FCL}_{out}(\tilde{\mathbf{y}}_{t+l_p}^f) \mathbf{w}_{out, b_{out}}) \quad (20)$$

where $\hat{\mathbf{y}}_{t+l_p} \in \mathbb{R}^{K \times l_y}$

The next time step of backward *RNN* is $t + l_p - 1$ because it is going backwards. All the steps from equation 16 to equation 20 are repeated for this time step $t + l_p - 1$ but this time the input hidden state is $h_{t+l_p}^{d-backward}$ and the output predicted trajectory vector is $\hat{\mathbf{y}}_{t+l_p-1}$. It will go backward till the time step $t + 1$ so that we have a set of output predicted trajectories for each time step of the prediction horizon.

$$\hat{\mathbf{Y}}_t = \{\hat{\mathbf{y}}_{t+1}, \hat{\mathbf{y}}_{t+2}, \dots, \hat{\mathbf{y}}_{t+l_p}\} \quad (21)$$

where $\hat{\mathbf{Y}}_t$ is a set of l_p predicted horizon time steps where each time step is K stochastic multimodal trajectory predictions of dimension l_y for the observation time step t . For a single stochastic trajectory prediction:

$$\hat{\mathbf{Y}}_t^k = \{\hat{\mathbf{y}}_{t+1}^k, \hat{\mathbf{y}}_{t+2}^k, \dots, \hat{\mathbf{y}}_{t+l_p}^k\} \quad (22)$$

where $k \in [0, K - 1]$ which represents k^{th} stochastic trajectory predictions for the observation time step t .

$\hat{\mathbf{Y}}_t$ will be converted into a tensor $\in \mathbb{R}^{l_p \times K \times l_y}$.

This was one complete cycle for the first observation time step t i.e., for \mathbf{x}_t whose first step was in equation 4. Now we will repeat the same steps for the next observation time step $t + 1$ i.e., for \mathbf{x}_{t+1} whose first step will be in 4, and we will get $\hat{\mathbf{Y}}_{t+1} \in \mathbb{R}^{l_p \times K \times l_y}$. These all will be stacked in a tensor vertically so that when all the iterations on the observation sequence are completed from observation time step of t to $t+l_o-1$, we will have a tensor of predicted trajectories $\hat{\mathbf{Y}}_t \in \mathbb{R}^{l_o \times l_p \times K \times l_y}$ and if the batch size is more than 1, then $\hat{\mathbf{Y}}_t \in \mathbb{R}^{b \times l_o \times l_p \times K \times l_y}$ where b is the batch size. This is the final output of the model for one input sequence of observation time step t . At this point, a batch is completed and the *Loss* is calculated for the batch according to the equation 23 as explained in the following section.

3.3.6 Loss Function

We define *Loss* function as used in the baseline method [2] is defined as follows:

$$\mathbf{L}_{Total} = \min_{\forall k \in \mathcal{K}} RMSE(\hat{\mathbf{Y}}_t^k, \mathbf{Y}_t) + RMSE(\hat{\mathbf{Y}}_t^g, \mathbf{Y}_t) + KLD(Q_\phi(z|\mathcal{X}_t, Y_t), P_\nu(z|\mathcal{X}_t)) \quad (23)$$

where $\mathbf{Y}_t \in \mathbb{R}^{l_o \times l_p \times l_y}$ is the ground truth trajectory, $\hat{\mathbf{Y}}_t^k \in \mathbb{R}^{l_o \times l_p \times \mathcal{K} \times l_y}$ is the \mathcal{K} predicted trajectories, $\hat{\mathbf{Y}}_t^g \in \mathbb{R}^{l_o \times l_p \times l_y}$ is the stepwise estimated goals trajectory by the *SGE* module, $Q_\phi(z|\mathcal{X}_t, Y_t)$ is the recognition network which learns probability distribution of z conditioned on input sequence \mathcal{X}_t and ground truth trajectory sequence Y_t parametrized by ϕ , $P_\nu(z|\mathcal{X}_t)$ is the prior network which learns probability distribution of z conditioned on input sequence \mathcal{X}_t parametrized by ν .

1) First term is the root mean square error *RMSE*, between the \mathcal{K} predicted trajectories $\hat{\mathbf{Y}}_t^k \in \mathbb{R}^{l_o \times l_p \times \mathcal{K} \times l_y}$ and ground truth $\mathbf{Y}_t \in \mathbb{R}^{l_o \times l_p \times l_y}$ and then chooses that $k \in \mathcal{K}$ which gives the minimum RMSE.

$$\frac{1}{l_o \times l_p \times l_y} \min_{\forall k \in \mathcal{K}} \frac{\sum_{l=0}^{l_o-1} \sum_{m=0}^{l_p-1} \sum_{n=0}^{l_y-1} (\hat{\mathbf{Y}}_{t,l,m,n}^k - \mathbf{Y}_{t,l,m,n})^2}{!} \quad (24)$$

2) Second term is the root mean square error *RMSE* between the stepwise estimated goals trajectory $\hat{\mathbf{Y}}_t^g \in \mathbb{R}^{l_o \times l_p \times \mathcal{K} \times l_y}$ and ground truth $\mathbf{Y}_t \in \mathbb{R}^{l_o \times l_p \times l_y}$.

$$\frac{1}{l_o \times l_p \times l_y} \sum_{l=0}^{l_o-1} \sum_{m=0}^{l_p-1} \sum_{n=0}^{l_y-1} (\hat{\mathbf{Y}}_{t,l,m,n}^g - \mathbf{Y}_{t,l,m,n})^2 \quad (25)$$

3) Third term is the *Kullback – Leibler* Divergence score which is a measure of the difference between two probability distributions. It is included in the *Loss* function to train the prior network in such a way that it learns the input sequence distribution conditioned on output sequence as accurately as possible.

3.4 Evaluation Metrics

Generally, there are two most used metrics for the task of trajectory prediction:

1. **Average Displacement Error (ADE):** This is a measure of the deviation of the whole predicted trajectory from the whole ground truth trajectory.
2. **Final Displacement Error (FDE):** This measures the distance between the final point of the predicted and ground truth trajectory.

In this work, the following three types of metrics are used.

3.4.1 Mean Square Error (MSE)

Mean square error between the predicted and ground truth trajectory as ADE.

$$MSE = \frac{1}{N \times l_p} \sum_{n=0}^{N-1} \sum_{i=1}^{l_p} (d(\hat{y}_i - y_i))^2 \quad (26)$$

3.4.2 Center Mean Square Error (C_{MSE})

Center means square error, CMSE is the same as MSE but the points in this case are the center of the bounding boxes instead of the bounding boxes themselves.

3.4.3 Center Final Mean Square Error (CF_{MSE})

Center final means square error is a kind of FDE, but the final point is the center of the final bounding box of the predicted and the ground truth trajectory.

$$CF_{MSE} = \frac{1}{N} \sum_{n=0}^{N-1} d(\hat{y}_{l_p} - y_{l_p})^2 \quad (27)$$

Chapter 4: Experimentation

4.1 Implementation Details

The observation length is of 0.5s i.e., 15 frames and the prediction horizon is of 1.5s i.e., 45 frames. All bounding boxes used are normalized with respect to the image size. The datasets give the bounding boxes in the form of (x_1, y_1, x_2, y_2) where (x_1, y_1) is the left-top corner of the bounding box and (x_2, y_2) is the bottom-right corner of the bounding box (*ltbr*). But these are converted into (cx, cy, w, h) format where (cx, cy) is the centroid of the bounding box, w is the width and h is the height of the bounding box (*cxcywh*) in order to have a more meaningful representation of spatial points. We use the same rule for both *JAAD* [4] and *PIE* [3] datasets. For the feature selection task, the same architecture as in the baseline method *SGNet – ED* [2] has been used. Due to constrained resources, the training was done at a smaller batch size than the baseline method. So, in order to have a fair comparison, the baseline method was also trained at the same batch size on which the proposed method has been trained i.e., now *test bench* is the same for both and the results of this work can be compared with the *reproduced results* of the baseline method for the conducted experiments.

4.2 Machine Setup

Machine Learning needs special hardware and software tools. Table 4.1 shows the machine setup used for the experiments.

Table 4.1: Machine Setup (*hardware + software*)

Hardware	Motherboard	MSI x570
	CPU	AMD Ryzen 9 5900X 12-Core Processor
	GPU	NVIDIA Quadro RTX 5000
	RAM	64 GB
	SSD	500 GB
	OS	Ubuntu 20.04
Software	Language	Python 3.6.12
	IDE	pytorch 1.7.1
	Library	64 GB
	GPU Computing Software	cudatoolkit 11.0.221
	Tool	

4.3 Feature Selection Experiments

4.3.1 Exploration Analysis

We first explore three different groups of features 1) contextual, 2) appearance and 3) attributes to do the analysis of their impact on the evaluation metrics of the model. In all three experiments, bounding box is *constantly* present as first

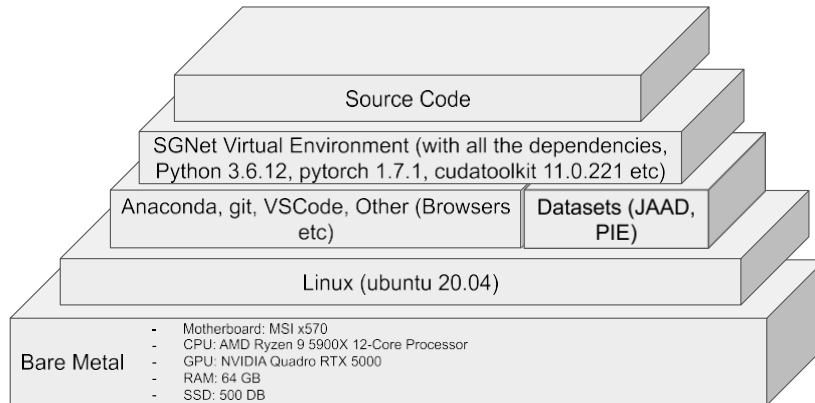


Figure 4.1: Hardware and software stack used for the experiments.

four dimensions of the input feature vector.

4.3.2 Comparative Analysis

Then best performing selected features are compared with the baseline method. These are compared with *reproduced results* of baseline method to have a fair comparison.

4.4 Bi-Directional RNN Decoder Experiments

RNNs suffer from accumulated error. So, a bi-directional decoder was proposed as explained in Section 3.3.5. So, the experiments were carried out to see the impact of bi-directional decoder in comparison with the baseline method.

Chapter 5: Results and Discussion

5.1 Reproduced Results of Baseline Method

Since the smaller batch size was used to reproduce the results of the baseline method, therefore the reproduced results are different from the published results of the baseline. The reproduced results are lower (lower the better) because decreasing the batch size makes the results better, lower in this case. The results are shown in the following table 5.1.

Table 5.1: Reproduced Results of the Baseline Method on the Test Bench (*↓ lower the better*)

<i>JAAD</i>					<i>PIE</i>				
<i>MSE ↓</i>			<i>C_{MSE} ↓</i>	<i>CF_{MSE} ↓</i>	<i>MSE ↓</i>			<i>C_{MSE} ↓</i>	<i>CF_{MSE} ↓</i>
0.5 s	1.0 s	1.5 s	1.5 s	1.5 s	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s
25	61	102	95	276	9	24	46	38	121

Note: These reproduced results are used as a *new* baseline for the proposed methods.

5.2 Feature Selection

5.2.1 Exploration Analysis

Table 5.2 shows the results of the exploration analysis. The best performing is attributes, appearance also improves but contextual is actually degrading the results.

This is because:

1) **Attributes** are capturing the realistic dependencies between the input features and the future trajectory. We, propose that age and gender are *good* predictors for predicting the future trajectory. It means that the trajectory behavior of different genders and different age groups are different, and the proposed model is successfully able to learn these different patterns.

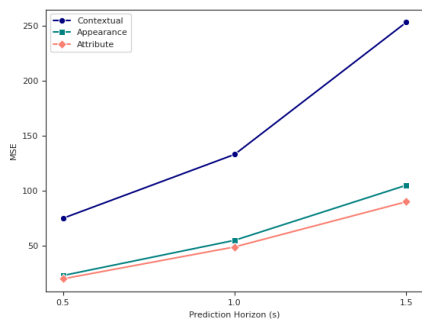
2) **Appearance** are basically redundant features. If we are taking bounding boxes in form of (cx, cy, w, h) where cx, cy represent the centroid of bounding boxes, w represents width and h represents height and then take into account the fact that we are using sequential data as input it becomes clear that the information of *pose_front*, *pose_back*, *pose_right* and *pose_left* are *implicit* already in the sequential data of the bounding boxes as: a) *pose_front*: h is increasing in the input temporal sequence, b) *pose_back*: h is decreasing in the input temporal sequence, c) *pose_right*: cx, cy is moving right in the input temporal sequence, and d) *pose_left*: cx, cy is moving left in the input temporal sequence. So, although these add some information, but not *significant*.

3) **Contextual** is in fact acting as a random noise for most of the pedestrians and also because it is present in a very small number of frames. It is acting as a random noise because in the datasets, the contextual signs and traffic signal's states are not associated with *unique* pedestrians. For example, a pedestrian can be present in a frame where the traffic signal is red, but he is going in a different

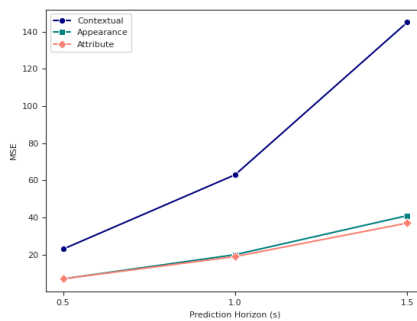
direction while the specific traffic signal is for some other path. Similarly, for signs that are for a target pedestrian, the sign is not relevant. But for a few pedestrians, these are relevant. So, this acts as *random noise* that sometimes is relevant, and sometimes it is not.

Table 5.2: Comparison of Selected Features (\downarrow lower the better)

Selected Features	<i>JAAD</i>					<i>PIE</i>				
	<i>MSE</i> \downarrow			<i>C_{MSE}</i> \downarrow		<i>MSE</i> \downarrow			<i>C_{MSE}</i> \downarrow	
	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s
Contextual	75	133	253	211	578	23	63	145	132	283
Appearance	23	55	105	99	257	7	20	41	36	109
Attributes	20	49	90	81	211	7	19	37	29	93



(a) *JAAD*



(b) *PIE*

Figure 5.1: Line graph comparing the results of different sets of selected features on *JAAD* and *PIE* datasets. Only the evaluation metric *MSE* for prediction horizon 0.5s, 1.0s and 1.5s is shown. Bounding box is present in all the sets of selected features.

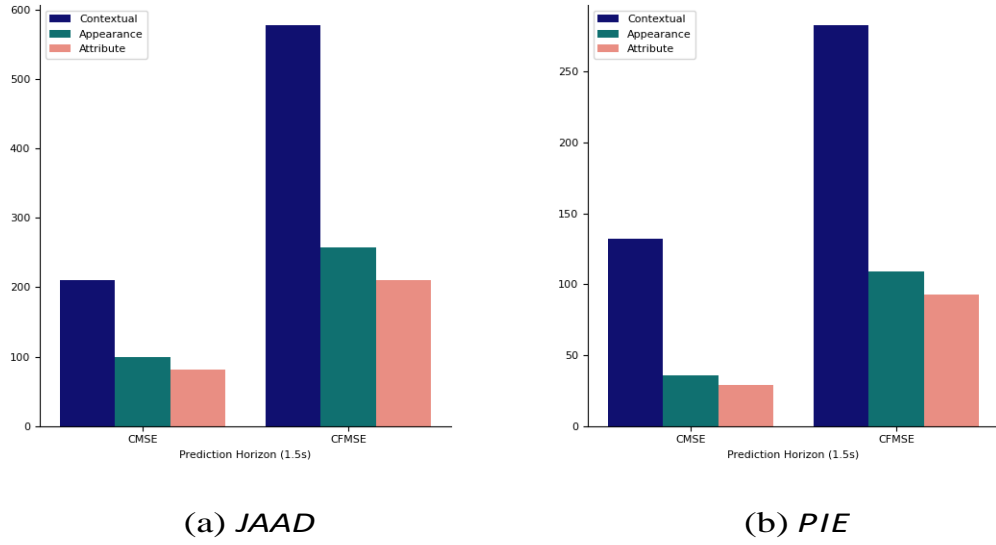


Figure 5.2: Bar graph comparing the results of different sets of selected features on *JAAD* and *PIE* datasets. Evaluation metric $CMSE$ and $CFMSE$ for prediction horizon of 1.5s is shown. Bounding box is present in all the sets of selected features.

5.2.2 Comparative Analysis

Table 5.3 shows the comparison of the best performing selected features with the baseline method. We compare it with *reproduced results* of the baseline method to have a fair comparison.

Table 5.3: Comparison of the Best Selected Features with the Reproduced Results of the Baseline Method SGNet-ED [2] on *JAAD* and *PIE* Datasets (\downarrow lower the better)

Method (Best of 20)	<i>JAAD</i>					<i>PIE</i>				
	<i>MSE</i> \downarrow			<i>C_{MSE}</i> \downarrow	<i>CF_{MSE}</i> \downarrow	<i>MSE</i> \downarrow			<i>C_{MSE}</i> \downarrow	<i>CF_{MSE}</i> \downarrow
	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s	0.5 s	1.0 s	1.5 s	1.5 s	1.5 s
SGNet-ED [2]	25	61	102	95	276	9	24	46	38	121
SGNet-AT	20	49	90	81	211	7	19	37	29	93

Note: The implementation details of reproduced results and the results of this work are the same, as mentioned in Section 5.1

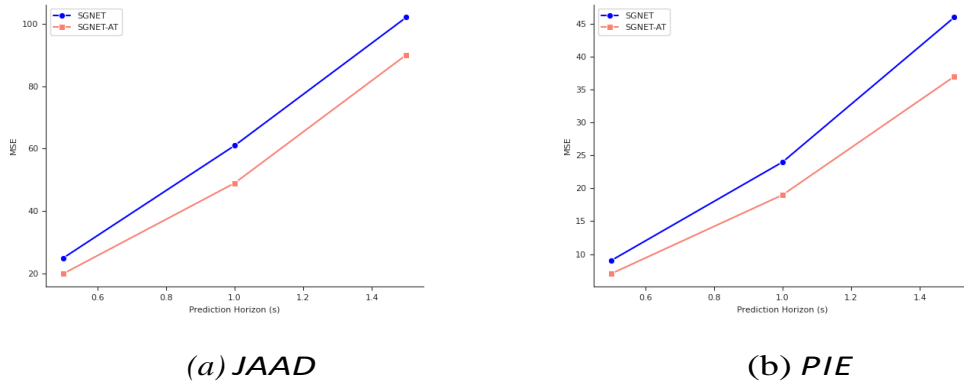


Figure 5.3: Graph comparing the results of the best selected features on *JAAD* and *PIE* datasets with reproduced results of the baseline method SGNet-ED [2].

5.3 Bi-Directional RNN Decoder

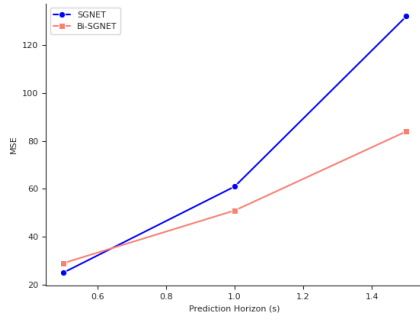
Table 5.4 compares the results of the proposed bi-directional decoder with the results of the baseline method SGNet-ED [2] on *JAAD* and *PIE* datasets. It

Table 5.4: Comparison of the Proposed Bi-Directional Decoder Bi-SGNet, with the Reproduced Results of the Baseline Method SGNet-ED [2] on *JAAD* and *PIE* Datasets (\downarrow lower the better)

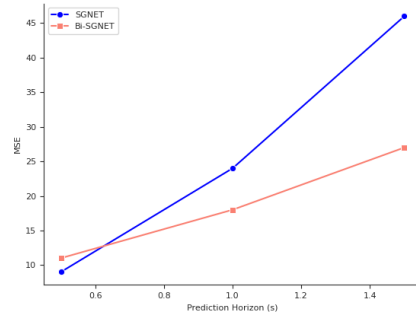
Method (Best of 20)	<i>JAAD</i>			<i>PIE</i>		
	<i>MSE</i> \downarrow			<i>MSE</i> \downarrow		
	0.5 s	1.0 s	1.5 s	0.5 s	1.0 s	1.5 s
SGNet-ED [2]	25	61	102	9	24	46
Bi-SGNet	29	51	76	11	18	29

Note: The Implementation details of reproduced results and the results of this work are same as mentioned in Section 5.1.

clearly shows that there is an improvement, but the future horizon of 0.5s is basically degrading, while the future horizon of 1.0s and 1.5s are improving. This is because the backward RNN is also suffering from the accumulated error problem but now in the backward direction i.e., from time step $t + l_p$ to time step $t + 1$ and since the output of both forward and backward *GRU* has been fused through *regressor layers*, there is some sort of *averaging* going on. But, this is also *weighted* as the *Loss* function is punishing the backward RNN's initial time steps more, simply because of the ground truth of these time steps.



(a) *JAAD*

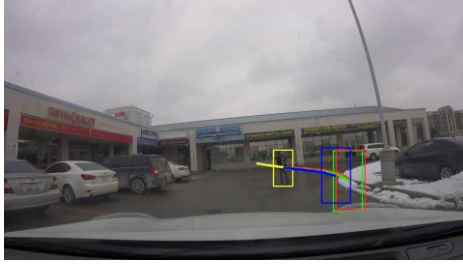


(b) *PIE*

Figure 5.4: Graph comparing the results of the proposed method Bi-SGNet with the reproduced results of the baseline method SGNet-ED [2] on *JAAD* and *PIE* datasets. This shows that the proposed method outperforms the baseline method in long term trajectory prediction of pedestrians on both datasets.

5.4 Qualitative Results

The following Figure 5.5 shows the qualitative results of the proposed method of bi-directional decoder on *JAAD* and *PIE* datasets in comparison with the baseline method. The proposed method is represented by green color and the baseline method is represented by blue color. It can be seen clearly that the bi-directional decoder based proposed method, Bi-SGNet is more accurate in long term predictions.



(a)



(b)

Figure 5.5: Qualitative results of the proposed method on *JAAD* and *PIE* datasets. **(a)** *JAAD* dataset, **(b)** *PIE* dataset. **Yellow** represents observed trajectory, **Red** represents future ground truth trajectory, **Blue** represents predicted trajectory by the baseline method, **Green** represents predicted trajectory by the proposed method of bi-directinal *RNN* decoder (*Bi-SGNet*).

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Through the experimentation and the results, following conclusions can be drawn for the research questions posed in the introduction:

1. Attributes (age and gender) features when added with spatial features improve the accuracy of the trajectory prediction of pedestrians for First Person View (FPV) ego-vehicle-centric camera sensor autonomous vehicles.
2. Bi-directional RNN decoder improves the accuracy of long term trajectory prediction.

Moreover, a novel bi-directional RNN decoder has been proposed. Which is also a contribution of this thesis work.

6.2 Future Work



Figure 6.1: Failure cases. **Yellow** represents observed trajectory, **Red** represents the ground truth trajectory, **Green** represents predicted trajectory. **(a)** Failure because of lack of interaction information. **(b)** Failure because of lack of roads and lanes information.

There are many more features present in both *JAAD* [4] and *PIE* [3] datasets which can be explored. There are many features which are present only in *PIE* dataset like ego-vehicle odometry and GPS data, which can be explored.

There are many failure cases which are because of the interaction between a target pedestrian and other road users, see Figure 6.1. So, interaction features can be explored, for which the interaction has to be represented by some appropriate representation.

Similarly, the information of roads and lanes in the scene plays an important role and there are failure cases which basically are because that current method doesn't take into account the information of roads and lanes.

Optical flow is a very important feature which tells the direction of motion. It should be explored to see its effect on the accuracy of trajectory prediction.

Similarly, for motion, velocity and acceleration features can be extracted and experiments can be done to see their effect on the accuracy of trajectory prediction. This work is for trajectory prediction of pedestrians for First Person View (FPV) ego-vehicle-centric camera sensor autonomous vehicles. It is a research question whether this method also works for other datasets like, where targets are vehicles, or for a multimodal full suite of sensors system, or for bird's eye view (BEV) datasets.

References

- [1] Sae international releases updated visual chart for its “levels of driving automation” standard for self-driving vehicles. [https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles](https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles). Accessed: 21 Jul, 2023.
- [2] Chuhua Wang, Yuchen Wang, Mingze Xu, and David J. Crandall. Step-wise Goal-Driven Networks for Trajectory Prediction. *IEEE Robotics and Automation Letters*, 7(2):2716–2723, April 2022.
- [3] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John Tsotsos. PIE: A Large-Scale Dataset and Models for Pedestrian Intention Estimation and Trajectory Prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6261–6270, Seoul, Korea (South), October 2019. IEEE.
- [4] Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos. Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 206–213, Venice, Italy, October 2017. IEEE.
- [5] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *Journal of Modern Transportation*, 24(4):284–303, December 2016.

- [6] R.E. Fenton and R.J. Mayhan. Automated highway studies at the Ohio State University-an overview. *IEEE Transactions on Vehicular Technology*, 40(1):100–113, February 1991.
- [7] Petros A. Ioannou, editor. *Automated Highway Systems*. Springer US, Boston, MA, 1997.
- [8] Donald Shoup Donald Shoup. The High Cost of Free Parking. *Journal of Planning Education and Research* 17:3-20, January 1997.
- [9] Asif Faisal, Tan Yigitcanlar, Md. Kamruzzaman, and Graham Currie. Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy. *Journal of Transport and Land Use*, 12(1), January 2019.
- [10] Anne Stockem Novo, Martin Krüger, Marco Stolpe, and Torsten Bertram. A Review on Scene Prediction for Automated Driving. *Physics*, 4(1):132–159, February 2022.
- [11] Adithya Ranga, Filippo Giruzzi, Jagdish Bhanushali, Emilie Wirbel, Patrick Pérez, Tuan-Hung Vu, and Xavier Perotton. VRUNet: Multi-Task Learning Model for Intent Prediction of Vulnerable Road Users. *Electronic Imaging*, 32(16):109–1–109–10, January 2020.
- [12] Henry Alexander Ignatious, Hesham-El Sayed, and Manzoor Khan. An overview of sensors in Autonomous Vehicles. *Procedia Computer Science*, 198:736–741, 2022.
- [13] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Pollefeys. 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, 68:14–27, December 2017.

- [14] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, Seattle, WA, USA, June 2020. IEEE.
- [15] Abdelmoghit Zaarane, Ibtissam Slimani, Wahban Al Okaishi, Issam Atouf, and Abdellatif Hamdoun. Distance measurement system for autonomous vehicles using stereo camera. *Array*, 5:100016, March 2020.
- [16] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-Sensor Fusion in Automated Driving: A Survey. *IEEE Access*, 8:2847–2868, 2020.
- [17] Michal Taraba, Juraj Adamec, Matus Danko, and Peter Drgona. Utilization of modern sensors in autonomous vehicles. In *2018 ELEKTRO*, pages 1–5, Mikulov, May 2018. IEEE.
- [18] Di Feng, Christian Haase-Schutz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, March 2021.
- [19] Phillip Karle, Maximilian Geisslinger, Johannes Betz, and Markus Lienkamp. Scenario Understanding and Motion Prediction for Autonomous Vehicles—Review and Comparison. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):16962–16982, October 2022.
- [20] Sajjad Mozaffari, Omar Y. Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction

- for Autonomous Driving Applications: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):33–47, January 2022.
- [21] Rim Trabelsi, Redouane Khemmar, Benoit Decoux, Jean-Yves Ertaud, and Rémi Butteau. Recent Advances in Vision-Based On-Road Behaviors Understanding: A Critical Survey. *Sensors*, 22(7):2654, March 2022.
- [22] Jurgen Wiest, Matthias Hoffken, Ulrich Kresel, and Klaus Dietmayer. Probabilistic trajectory prediction with Gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium*, pages 141–146, Alcal de Henares , Madrid, Spain, June 2012. IEEE.
- [23] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, Las Vegas, NV, USA, June 2016. IEEE.
- [24] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. Probabilistic Prediction of Vehicle Semantic Intention and Motion. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 307–313, Changshu, June 2018. IEEE.
- [25] Stefan Zernetsch, Sascha Kohnen, Michael Goldhammer, Konrad Doll, and Bernhard Sick. Trajectory prediction of cyclists using a physical model and an artificial neural network. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 833–838, Gotenburg, Sweden, June 2016. IEEE.
- [26] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-Agent Tensor Fusion for Contextual Trajectory Prediction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12118–12126, Long Beach, CA, USA, June 2019. IEEE.

- [27] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, Salt Lake City, UT, June 2018. IEEE.
- [28] Florin Leon and Marius Gavrilescu. A Review of Tracking and Trajectory Prediction Methods for Autonomous Driving. *Mathematics*, 9(6):660, March 2021.
- [29] Cinzia Viroli and Geoffrey J. McLachlan. Deep Gaussian mixture models. *Statistics and Computing*, 29(1):43–51, January 2019.
- [30] Nachiket Deo and Mohan M. Trivedi. Convolutional Social Pooling for Vehicle Trajectory Prediction. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1549–15498, Salt Lake City, UT, USA, June 2018. IEEE.
- [31] Nachiket Deo and Mohan M. Trivedi. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184, Changshu, June 2018. IEEE.
- [32] Xinchun Yan, Jasmine Hsu, Mohi Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee. Learning 6-DOF grasping interaction via deep 3D geometry-aware representations. In *Proceedings of (ICRA) international conference on robotics and automation*, pages 3766 – 3773, May 2018.
- [33] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S. Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks. In *Proceedings of the 33rd international conference on neural information pro-*

cessing systems. Curran Associates Inc., Red Hook, NY, USA, 2019. Number of pages: 10 tex.articleno: 13.

- [34] Zhenning Li, Zhiwei Chen, Yunjian Li, and Chengzhong Xu. Context-aware trajectory prediction for autonomous driving in heterogeneous environments. *Computer-Aided Civil and Infrastructure Engineering*, page mice.12989, March 2023.
- [35] Sarthak Sharma, Junaid Ahmed Ansari, Krishna Murthy Jatavallabhula, and K. Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3508–3515, 2018.
- [36] Nicholas Rhinehart, Rowan Mcallister, Kris Kitani, and Sergey Levine. PRECOG: PREDiction Conditioned on Goals in Visual Multi-Agent Settings. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2821–2830, Seoul, Korea (South), October 2019. IEEE.
- [37] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofghi, and Silvio Savarese. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1349–1358, Long Beach, CA, USA, June 2019. IEEE.
- [38] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, Salt Lake City, UT, June 2018. IEEE.
- [39] Karl Pertsch, Oleh Rybkin, Frederik Ebert, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. In *Proceedings of the 34th international conference on neural information processing systems*, NIPS’20, Red Hook, NY, USA,

2020. Curran Associates Inc. Number of pages: 13 Place: Vancouver, BC, Canada tex.articleno: 1453.

- [40] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, Seattle, WA, USA, June 2020. IEEE.
- [41] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the neural information processing systems track on datasets and benchmarks (NeurIPS datasets and benchmarks 2021)*, 2021.
- [42] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 conference on robot learning*, volume 155 of *Proceedings of machine learning research*, pages 409–418. PMLR, November 2021.
- [43] Andrey Malinin, Neil Band, Yarin Gal, Mark Gales, Alexander Ganshin, German Chesnokov, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, Vyas Raina, Denis Roginskiy, Mariya Shmatova, Panagiotis Tigas, and Boris Yangel. Shifts: A dataset of real distributional shift across multiple large-scale tasks. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 2)*, 2021.
- [44] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Stanford drone dataset.

- [45] S Pellegrini, A Ess, K Schindler, and L Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, Kyoto, September 2009. IEEE.
- [46] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by Example. *Computer Graphics Forum*, 26(3):655–664, September 2007.
- [47] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on computer vision and pattern recognition (CVPR)*, 2012.
- [48] U.S. Department Of Transportation Federal Highway Administration. Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data, 2017.
- [49] Bhakti A Paranjape and Apurva A Naik. DATS_2022: A Versatile Indian Dataset for Object Detection in Unstructured Traffic Conditions. *Data in Brief*, 43:108470, August 2022.
- [50] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021. Publisher: SAGE Publications Sage UK: London, England.