ADVANCING HIGHWAY SAFETY: EMBEDDED-EDGE AI FOR REAL-TIME APPLICATIONS

by

Vinit Amrutlal Katariya

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2023

Approved by:

Dr. Hamed Tabkhi

Dr. Omidreza Shoghli

Dr. James Conrad

Dr. Dipankar Maity

Dr. Yuanan Diao

Copyright Notes

[1] http://www.ieee.org/publications_standards/publications/rights/rights_link.html

ABSTRACT

VINIT AMRUTLAL KATARIYA. Advancing Highway Safety: Embedded-edge AI for Real-time Applications. (Under the direction of DR. HAMED TABKHI)

This dissertation presents the systematic design and development of datasets, algorithms, and an AI pipeline specifically curated for real-time trajectory prediction and anomaly detection in highway environments. These innovations are meticulously optimized for embedded-edge systems, ensuring timely outputs for safety and surveillance applications.

First, the dissertation presents **DeepTrack: Lightweight Deep Learning for Vehicle Path Prediction in Highways**, a deep learning model tailored for edge systems. This model uniquely employs Agile Temporal Convolutional Networks (ATCNs) rather than the traditionally-used Long Short-Term Memory (LSTM) networks to encapsulate vehicle dynamics. Not only does DeepTrack boast equivalent or superior accuracy to leading trajectory prediction models, but it also shines in its diminished model size and reduced computational intensity, making it ideal for embedded edge systems. Distinctly, vehicle interactions are interpreted through ATCNs instead the frequently-associated LSTM in time series analysis. A hallmark of ATCN is its depthwise convolution, which significantly curtails model complexity in comparison to LSTMs, both in size and operational demands. Experimental results show that DeepTrack cuts Average Displacement Error (ADE) by 12.23%, reduces the Final Displacement Error (FDE) by 2.69%, and also decreases both the number of operations and the model size by approximately 21.67% and 43.13%, respectively, compared to then State of the Art (SotA) trajectory prediction algorithm.

Subsequently, **Carolina Highway Dataset (CHD)**, a unique highway trajectory dataset captured from two distinct Points Of View (POVs) â high-angle and eye-level is introduced. While numerous vehicle trajectory datasets exist, most lack the

diversity of driving scenes that capture various highway designs, merging lanes, and configurations. CHD, however, stands out by offering data from 1.6 million frames and 338,000 vehicle trajectories recorded in highway videos, encapsulating both eye-level and high-angle perspectives from eight strategically selected locations across the Carolinas. These locations, accompanied by meticulous timing and camera angles, ensure a comprehensive representation of road geometries, traffic trends, lighting variations, and diverse driving behaviors.

Additionally, **PishguVe** â a SotA vehicle trajectory prediction architecture that uses attention-based graph isomorphism and convolutional neural networks is presented next. When tested, PishguVe excelled by outperforming pre-existing SotA algorithms across bird's-eye, eye-level, and high-angle POV trajectory datasets. Notably, on the NGSIM dataset, it achieved a commendable improvement of 12.50% in Average Displacement Error (ADE) and 10.20% in Final Displacement Error (FDE) compared to the current SotA. Against the top-performing models on CHD, PishguVe demonstrated superior results, reducing the ADE and FDE on eye-level as well as high-angle data.

The final contribution details **VegaEdge: A Confluence AI Approach for Video Anomaly Detection at the Edge in Real-Time Highway Safety**. It commences with the introduction of the **Carolinas Anomaly Dataset (CAD)**, aiming to fill the prevalent void in datasets specifically designed for highway anomalies. The significance of vehicle anomaly detection is underscored, emphasizing its pivotal role in various highway safety applications, including accident prevention, rapid response, traffic flow optimization, and work zone safety. In alignment with this vision, a novel lightweight technique for vehicle anomaly detection is put forth, harnessing the prowess of trajectory prediction. The proposed methodology adeptly spots vehicles that deviate from anticipated paths, pinpointing potential highway risks across diverse camera perspectives, and utilizing real-world highway datasets.

Further, the edge detection framework, **VegaEdge** is introduced. It represents a refined AI confluence that strategically chooses agile algorithms and techniques. This selection is adept at tasks ranging from detection and tracking to trajectory prediction and anomaly identification in real-time, catering to contemporary security and surveillance needs in modern highways via edge-focused IoT-embedded platforms. In addition, VegaEdge, when deployed on an embedded IoT platform, efficiently processes 738 trajectories per second in standard highway environments, illustrating its adaptability and efficiency.

DEDICATION

To my dear mother, whose love and enduring belief in me shine as the guiding light in all my endeavors. Though she may not be with us today, her memories, teachings and ideals will always be with me in every milestone I achieve.

Foremost, to my loving wife, Rasika. The challenges of this Ph.D. journey would have been impossible without her relentless support, patience, and unwavering love. Her faith in my abilities has been the cornerstone of my resilience and determination throughout this voyage, and for that, I am forever grateful.

I extend my heartfelt appreciation to my father, Amrutlal Bansilal Katariya, who has consistently been a pillar of strength and support. My gratitude is also directed towards my uncle, Jivan Katariya, and aunt, Sunita Katariya, for their unceasing guidance and wisdom. My siblings, Komal, Kaushal, and Pooja, have immensely enriched my life, showering me with affection, laughter, and invaluable insights. I am deeply grateful to my father-in-law, Adv. Chandrashekhar Dharne, mother-in-law, Sujata Dharne, and Mohini, who have ceaselessly bolstered my spirits. To my nephew Tushar, a source of boundless joy and optimism in our family. I'd like to extend my gratitude to Jayesh Desai for guiding me through the challenging moments of this journey. Additionally, my heartfelt thanks go to my friends Ajinkya Mane and Hrishikesh Tamhane for their love and support.

As this chapter concludes, it announces the onset of a new journey. Equipped with the lessons from the past and the blessings of my loved ones, I step forward with anticipation and hope, eager to embrace what the future holds.

ACKNOWLEDGEMENTS

My deepest appreciation goes to my advisor, Dr. Hamed Tabkhi. Entrusting me with the opportunity to work under his expert guidance, he has consistently exhibited faith in my abilities. His unwavering support, profound insights, and exceptional mentorship have been instrumental throughout this endeavor. His dedication to nurturing a progressive research atmosphere and his genuine belief in my potential were pivotal in realizing this dissertation. Beyond honing my skills as a researcher and engineer, his guidance has also enriched me personally, shaping me into a more discerning individual.

I extend my heartfelt gratitude to Dr. Omidreza Shoghli for his invaluable guidance on the intricacies of transportation and work zone applications. His keen insights and thoughtful recommendations have greatly enriched my research journey. I would like to extend my sincere gratitude to my dissertation committee members: Dr. James Conrad, Dr. Dipankar Maity, and Dr. Yuanan Diao, for their patience, invaluable feedback, and encouragement.

My journey at UNC-Charlotte has been enriched by the camaraderie and support of my peers. Special thanks to Dr. Mohammadreza Baharani, Armin Danesh Pazho, Ghazal Alinezhad Noghre, Chris Neff, Babak Rahimi Ardabili, Eric Yao, and Narges Rashvand. The collective spirit and collaborative atmosphere at the TeCSAR Lab have been pivotal to my academic and personal growth.

TABLE OF CONTENTS

LIST OF TABLES xii

LIST OF FIGURES xiv

LIST OF ABBREVIATIONS xvii

CHAPTER 1: INTRODUCTION 1

1.1. Motivation 1

1.2. Contributions to the Body of Knowledge 3

CHAPTER 2: DEEPTRACK: LIGHTWEIGHT DEEP LEARNING FOR VEHICLE TRAJECTORY PREDICTION IN HIGHWAY 6

2.1. Introduction 6

2.2. Related Work 8

2.3. Motivation: Applications of Real-time Trajectory Prediction in Highways 10

2.4. DeepTrack Design 11

2.4.1. Model Architecture 12

2.4.2. Preliminaries and Problem Formulation 13

2.4.3. Algorithm 19

2.5. Evaluation 20

2.5.1. Evaluation Metrics 22

2.5.2. DeepTrack Models and Comparisons 23

2.5.3. Comparison Against Existing Approaches 26

2.5.4. Qualitative Results 28

2.6. Conclusion and Future Works 30

CHAPTER 3: A POV-BASED HIGHWAY VEHICLE TRAJECTORY DATASET AND PREDICTION ARCHITECTURE 31

3.1. Introduction 31

3.2. Related Works 34

3.2.1. Vehicle Tracking and Trajectory Datasets 34

3.2.2. Vehicle Trajectory Prediction Algorithms 35

3.3. Carolinas Highway Dataset (CHD) 37

3.3.1. Annotations and Trajectory Extraction 38

3.3.2. CHD Statistics and Format 39

3.4. PishguVe 40

3.4.1. Network Architecture 42

3.5. Evaluation and Experiments 44

3.5.1. Evaluation Metrics 45

3.5.2. Hyperparameter Search 47

3.5.3. Results 48

3.6. Conclusion 51

CHAPTER 4: VEGAEDGE: EDGE AI CONFLUENCE ANOMALY DETECTION FOR REAL-TIME HIGHWAY IOT-APPLICATIONS 53

4.1. Introduction 53

4.2. Related Works 56

4.3. CAD: Carolinas Anomaly Dataset 58

4.4. Anomaly detection Methodology 59

4.4.1. ADE-based Anomaly Detection 59

4.4.2. Angle-based Anomaly Detection 60

4.5. VegaEdge Design 60

4.5.1. Vehicle Detection 63

4.5.2. Vehicle Tracking 63

4.5.3. Data Conditioning 64

4.5.4. Trajectory Prediction 65

4.5.5. Prediction-based Anomaly Detection 65

4.5.6. VegaEdge Performance Benchmarking 66

4.6. Results 67

4.6.1. Anomaly detection on NGSIM Dataset: 67

4.6.2. Anomaly detection on CAD 69

4.6.3. Real-world vs Adversarial Anomaly Trajectories 71

4.6.4. VegaEdge Performance 73

4.6.5. Highway Workzone Safety Application 74

4.6.6. VegaEdge Power Analysis on IoT Platform 76

4.7. Conclusion 76

CHAPTER 5: CONCLUSIONS 78

REFERENCES 81

LIST OF TABLES

TABLE 2.1: DeepTrack Variants (Transposed) 21

TABLE 2.2: ATCN encoder configuration for ego vehicle and neighbors. 23

TABLE 2.3: Performance comparison of DeepTrack variants based on Root mean square error, Displacement errors, and Computation. 24

TABLE 2.4: Performance comparison of best DeepTrack variant DT_2 based on different data splits of NGSIM dataset. RMSE, FDE, ADE are in meters. 26

TABLE 2.5: Prediction and Model complexity comparison of DeepTrack with Trajectory forecasting models based on NGSIM dataset. 26

TABLE 3.1: Comparison of existing trajectory datasets with CHD. T and BB in the Annotation column stand for Trajectories and Bounding Boxes (BB) respectively. FPS stands for Frames Per Second. BE stands for Bird's-eye-view, DC stands for Dashcam view, HA and EL stand for High-angle and Eye-level view respectively. 33

TABLE 3.2: Comparison of existing camera view vehicle detection and tracking datasets with CHD. In the function column, D is for Detection, TP is for Trajectory Prediction, AD is for Accident Detection, and T is for Tracking. FPS stands for Frames Per Second. BE stands for Bird's-eye-view, DC stands for Dashcam view, HA and EL stand for High-angle and Eye-level view respectively. 36

TABLE 3.3: Performance comparison of vehicle trajectory prediction approaches on NGSIM datasets[1, 2]. 46

TABLE 3.4: Average and final displacement error-based performance of PishguVe with varying dropout probabilities. 47

TABLE 3.5: Performance comparison of vehicle trajectory prediction approaches on CHD eye-level POV data. 48

TABLE 3.6: Performance comparison of vehicle trajectory prediction approaches on CHD high-angle POV data. 50

TABLE 4.1: Accuracy comparison of each algorithm. 66

TABLE 4.2: EER for various Time Thresholds using ADE-and Angle based Anomaly on NGSIM Adversarial trajectories 69

TABLE 4.3: EER for various Time Thresholds using ADE and Angle based Anomaly on CAD 73

TABLE 4.4: Throughput (processed trajectories per second) comparison of VegaEdge on the three platforms across different road types and traffic densities (expressed as vehicle trajectories processed per second) 74

TABLE 4.5: Comparison of Anomaly detection Prediction Windows, Buffer Times (Excluding 3s Input Trajectory), and Vehicle Distances from Camera at 60 mph. 75

TABLE 4.6: Comparison of VegaEdge Power consumption and Throughput ((Trajectories processed per second) on Jetson Orin Power Modes for real word highway ($\sim$ 140 Vehicles detected per second across 30 frames.). Power is measured in Watts. 75

LIST OF FIGURES

FIGURE 2.1: Structure of TCN based DeepTrack encoder. 12

FIGURE 2.2: The path prediction model workflow. The location of the neighbors (gray triangles) and car of interest (solid red triangle) is shown at t_0 in Vehicle trajectory data (extreme left) block and Trajectory prediction (extreme right) block. Triangles denoting semi-transparent red in Vehicle trajectory data block, and semi-transparent blue in Trajectory prediction block represent observed history paths, and model output respectively. The observed history paths of neighbours (for past 3 seconds) are used by the model but not shown in the figure to avoid confusion. 15

FIGURE 2.3: Additive Attention. The output of softmax is multiplied with DeepTrack encoder output to lead the next layers on essential features. 18

FIGURE 2.4: Congested traffic 28

FIGURE 2.5: Lane-keeping 28

FIGURE 2.6: Maneuvering - Passing from left 28

FIGURE 2.7: Maneuvering - Passing from right 28

FIGURE 2.8: Congested traffic scenario: the location of the neighbors (gray triangles) is shown at t_0. Triangles denoting red, green, and blue respectively, represent observed history paths, ground truth, and model output. 28

FIGURE 2.9: Congested traffic 29

FIGURE 2.10: Multiple lane changing 29

FIGURE 2.11: Two instances where the model could not accurately predict the trajectory due to erratic driver behavior. The legend remains consistent with Fig. 2.8 29

FIGURE 3.1: Spatial distribution of all the eight recording locations on the highways of NC and SC included in CHD. 37

FIGURE 3.2: Vehicle distribution in the CHD trajectory data across test, train, and validation sets. 40

FIGURE 3.3: PishguVe architecture overview. Input trajectory vector, V_i (where i $\in$ (1 to n)) and relative trajectory vector ,ΔV_i (where i $\in$ (1 to n)) with x and y coordinates and input time window of T_{in} are concatenated and inferenced through the fully connected layer. The output of the fully connected layer, N_i, and neighbor aggregation vector ΔN_i is utilized by Attentive GIN in two separate branches for aggregating the node and neighbor features for detailed representation. The spatiotemporal attentive CNN is then used to predict the future trajectories of all vehicles up to T_{out} time steps ahead. 41

FIGURE 3.4: The Linear Attention Dropout (LAD) Block. LAD is integrated into GIN for feature aggregation using channel and spatial attention. Dropout layers are added after linear and attention layers to minimize overfitting. 43

FIGURE 3.5: Error matrices visualization for two seconds time window and a data rate of five samples/second. 45

FIGURE 4.1: Example demonstration of VegaEdge implementation on an IoT platform for real-world highway scenarios. 54

FIGURE 4.2: Data flow and algorithmic design of VegaEdge. Object detector and tracking process the frames in a single-frame manner. The filtering algorithm ensures that the focus is on the correct side of the road, and the Collection algorithm collects the sufficient number of frames T_{in} required by the Trajectory Prediction algorithm. The predicted path for T_{out} frames in comparison with the actual path will be used for Anomaly Detection, which finally can be used for highway IoT applications at the edge. 61

FIGURE 4.3: Fine grain (0.2s step) 67

FIGURE 4.4: Coarse grain (1s step) 67

FIGURE 4.5: AUC-ROC with varying detection windows for ADE-based anomaly detection method on NGSIM dataset. (a) Shows plot from 0s to 1s with time-step of 0.2 s (b) Shows plot from 1s to 5s with time-step of 1.0 s 67

FIGURE 4.6: Fine grain (0.2s step) 68

FIGURE 4.7: Coarse grain (1s step) 68

FIGURE 4.8: AUC-ROC with varying detection windows for angle-based anomaly detection method on NGSIM dataset. (a) Shows plot from 0s to 1s with time-step of 0.2 s (b) Shows plot from 1s to 5s with time-step of 1.0 s 68

FIGURE 4.9: ADE-based 68

FIGURE 4.10: Angle-based 68

FIGURE 4.11: EER plots for anomaly detections with 1s trajectory window for NGSIM dataset adversarial trajectories: (a) ADE-based anomaly, and (b) Angle-based anomaly. 68

FIGURE 4.12: Fine grain (0.2s) 70

FIGURE 4.13: Coarse grain (1s) 70

FIGURE 4.14: AUC-ROC with varying prediction windows for ADE-based anomaly detection method on CAD. (a) Shows plot from 0s to 1s with a time-step of 0.2 s. (b) Shows plot from 1s to 5s with time-step of 1.0 s. 70

FIGURE 4.15: Fine grain (0.2s) 71

FIGURE 4.16: Coarse grain (1s) 71

FIGURE 4.17: AUC-ROC with varying prediction windows for Angle-based anomaly detection method on CAD. (a) Shows plot from 0s to 1s with a time-step of 0.2 s. (b) Shows plot from 1s to 5s with time-step of 1.0 s. 71

FIGURE 4.18: ADE-based 72

FIGURE 4.19: Angle-based 72

FIGURE 4.20: EER plots for 4s anomaly detections for CAD dataset: (a) ADE-based anomaly, and (b) Angle-based anomaly. 72

LIST OF ABBREVIATIONS

ADE Average Displacement Error

AO Always On

ATCN Agile Temporal Convolutional Network

AUC-ROC Area Under the Receiver Operating characteristic curve

BB Bounding Boxes

BE Birdâs-eye-view

BN Batch Normalization

BN Batch-Normalization

CAD Carolinas Anomaly Dataset

CHD Carolinas Vehicle Dataset

CNN Convolutional Neural Networks

CPS Cyber-Physical Systems

CPU Central Processing Unit

DC Dashcam view

DNN Deep Neural Network

DW depthwise

EER Equal Error Rate

EL Eye-level view

FARS Fatality Analysis Reporting System

FDE Final Displacement Error

FPS Frames Per Second

GCN Graph Convolutional Networks

GIN Graph Isomorphism Network

GMMs Gaussian Mixture Models

GNN Graph Neural Network

GPU Graphical Processing Unit

GRU Gated Recurrent Unit

HA High-angle view

I-80 Interstate-80

IoT Internet of Things

ITS Intelligent Transportation Systems

LAD Linear Attention Dropout

LR Learning Rate

LSTM Long Short-Term Memory

mAP Average Precision

MOT Multi-object tracking

NGSIM Next generation simulation

NHTSA National Highway Traffic Safety Administration

POV Point-of-View

PW pointwise

ReLU Rectified Linear Unit

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SOC System-on-Chip

SotA State of the Art

STCB Spectral-Temporal Convolution Block

T Trajectory

TCN Temporal Convolutional Networks

TTC Time-to-Collision

V2I Vehicle-to-Infrastructure

V2V vehicle-to-vehicle

VIAC Vehicular Interactive Aware Convolution

CHAPTER 1: INTRODUCTION

1.1 Motivation

Modern transportation systems, driven by Intelligent infrastructure and smarter vehicles, are constantly moving towards safer and more efficient highways. As the emphasis on Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications grows, the quest for an end-to-end predictive solution for highway safety becomes paramount. While there have been significant advances in communication systems and trajectory prediction, alarming statistics still underscore the severity of highway-based accidents [3]. Highway-based accidents and fatalities remain a serious concern for transportation agencies and the public. In 2019, the United States reported 36,096 fatalities on its roadways [3, 4]. The National Highway Traffic Safety Administration documented another surge to 42,000 fatalities in 2022, with significant proportions of these accidents taking place on highways [5]. Amidst these alarming statistics, there is a noticeable void in the field of end-to-end predictive solutions tailored for highway safety.

A closer inspection into the domain revealed a distinct lack of vehicle trajectory datasets with a specific focus on highways. The majority of available datasets provided constrained perspectives like bird's-eye views or low-resolution high-angle videos [1, 2, 6, 7, 8, 9, 10]. A deep dive into the existing literature and systems reveals that while vehicle trajectory prediction models have seen incremental growth in accuracy, reaching up to 90% for short-term predictions. Such models might not directly extrapolate to the peculiarities and unique requirements of highway systems. Furthermore, a majority of the prediction algorithms were dedicated to pedestrians or complex urban scenarios characteristic of autonomous driving, leaving a discernible gap in models

tailored for highway conditions [11, 12, 13, 14].

In a concerted effort to bridge this noticeable gap, DeepTrack was proposed as a novel solution designed specifically for real-time safety-critical applications. DeepTrack can provide precise Time-to-Collision (TTC) predictions, crucial for accident avoidance on highways. DeepTrack not only showcases high accuracy but does so with reduced computational complexity, making it apt for embedded systems at the edge, a necessary advancement for real-time applications [15, 16]. The introduction of Agile Temporal Convolutional Networks (ATCN) in DeepTrack offers a promising solution to the complexities associated with traditional LSTM-based models, marking a significant stride in trajectory prediction for highway safety. Additionally, understanding the importance of having a multi-perspective, comprehensive dataset for highway systems, the Carolinas Vehicle Dataset (CHD) was introduced. The CHD encapsulates a myriad of trajectories under different lighting and weather conditions, giving researchers a valuable resource to develop better predictive algorithms [17, 18, 19]. It also marks a step forward, offering a comprehensive repository capturing naturalistic highway driving behaviors from varied points of view [20, 21]. Such datasets can be pivotal in developing Intelligent Transportation Systems (ITS) tailored specifically for highway surveillance and safety.

Taking a step further into the interconnected world of the Internet of Things (IoT), there's a pressing need to harness real-time analytics from the massive influx of data from highway cameras. With the rising incidents, especially within work zones, the existing surveillance infrastructure demands the power of edge-based AI frameworks [22, 23]. Current anomaly detection systems, majorly tailored for urban setups, often do not cater to the specificities of highway scenarios [24]. Addressing this deficiency, we introduce the Carolinas Anomaly Dataset (CAD) for real-world highway anomalies, coupled with VegaEdge, a groundbreaking AI solution tailored for real-time highway applications [25]. VegaEdge not only offers rapid detection but also

scales to provide efficient highway monitoring, heralding a transformative approach to road safety and management. By operating on embedded edge devices, VegaEdge promises rapid detection while optimizing computational resources, ensuring safety in the ever-evolving highway environment.

In conclusion, the journey towards a safe and efficient highway system demands an intricate weaving of precise prediction models, comprehensive datasets, and agile edge-based AI solutions. With advances such as DeepTrack, CHD, CAD, and VegaEdge, we are progressively inching closer to this ideal, ensuring that our highways become safer conduits of transport for all.

1.2 Contributions to the Body of Knowledge

In the broader context of advancing highway safety and vehicular technologies, this dissertation delineates the major contributions that augment the existing literature in the following ways:

- Introduced a deep learning model specifically tailored for highway trajectory prediction, achieving high accuracy and reduced computational complexity. Incorporated Agile Temporal Convolutional Networks (ATCN) to overcome traditional LSTM-based model limitations. The model serves as a lightweight trajectory forecasting tool, predicting the vehicle of interest's location up to 5 seconds in the future. A novel encoder design was implemented, decreasing the complexity and size of the overall network by at least 22% and 12% respectively, in comparison to state-of-the-art trajectory prediction models, while maintaining comparable accuracy. An extensive analysis of the proposed network's design was provided, emphasizing the influence of various components on performance.

- This work presents a comprehensive dataset CHD capturing naturalistic highway driving behaviors. CHD, is a robust vehicle trajectory dataset sourced

from multiple POVs, capturing diverse vehicle maneuvers on highways, ranging from varied geometries to lane mergers. This dataset fills a critical niche by offering high-quality data for highway environments and aids in the development of Intelligent Transportation Systems (ITS) with a focus on highway safety. Complementing this, CHD provides 1.6 million high-resolution images enriched with annotation data, facilitating advanced detection and tracking applications. Alongside this, is presented PishguVe, a cutting-edge trajectory prediction network grounded on attention-based convolutional and graph isomorphism networks. PishguVe secures a best-in-class error rate across multiple datasets, showcasing its adaptability for real-time trajectory prediction across various POV scenarios. The utility and effectiveness of both CHD and PishguVe are highlighted by offering a detailed comparison of the trajectory datasets. Furthermore, an exhaustive evaluation of trajectory prediction algorithms, trained and evaluated in NGSIM and CHD, reinforces our claim of superiority and relevance in the field.

- To bridge the gap in highway-based anomaly datasets, the Carolinas Anomaly Dataset (CAD), is introduced to bolster research efforts in real-world highway settings. This dataset has 22 videos of real-world highway anomalies with vehicles deviating from the lanes, vehicles approaching the camera, and vehicles stopping in front of the camera. In conjunction, we propose a distinctive anomaly detection method that detects anomalous driving behaviors from predicted trajectories by calculating angle-based and displacement errors. The method's efficacy is proved using both adversarial and real-world trajectories across specific datasets. By doing this it highlights the effects of adversarial anomalies versus real-world anomalies with varying prediction windows. To further the cause, this dissertation also introduces VegaEdge, an avant-garde AI-driven IoT solution, adept at anomaly detection in vehicles, specifically en-

gineered for edge-based embedded systems. This system excels at pinpointing vehicles that stray from expected paths, serving as an early warning system for potential highway threats.

- VegaEdge, alongside our proposed anomaly detection methodologies, undergo rigorous evaluations spanning multiple platforms and scenarios. The outcome attests to its versatility and unmatched prowess in both actual and simulated conditions. VegaEdge's real-time processing of real-world anomalies is presented by calculating throughput for varying traffic densities on Nvidia Jetson Orin AGX and Jetson NX boards. Furthermore, its effectiveness is underscored, especially in enhancing work zone safety, as we present the changing buffer time with the prediction window used by the anomaly detection model.

In summation, this dissertation charts the evolution of trajectory prediction from its nascent stages, harnessing the power of ATCN and attention-based methodologies, to the culmination of a holistic AI solution dedicated to highway safety. This journey was marked by several key contributions, including the introduction of authentic real-world datasets and the development of state-of-the-art trajectory prediction algorithms. These incremental advancements, superior to prior efforts, created a new route for devising highway safety applications. Notably, the emphasis on embedded-edge devices acting in real-time has the potential to be transformative, heralding a future where countless lives and injuries can be safeguarded on our roads.

CHAPTER 2: DEEPTRACK: LIGHTWEIGHT DEEP LEARNING FOR VEHICLE TRAJECTORY PREDICTION IN HIGHWAY

2.1 Introduction

With the advent of high-speed communication systems and unprecedented improvements in trajectory predicting, we are closer to implementing a fully connected (vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I)) and fully-aware transportation system than ever before. The increasing push towards autonomous driving and thrust for designing the best in class crash-avoidance systems at the edge has resulted in development of trajectory prediction algorithms with better than 90% accuracy for up to 5 seconds in the future. The use of such high accuracy models in safety-critical systems for crash avoidance and accident prediction can result in precise time-to-collision (TTC) prediction [26], which can prove instrumental in avoiding accident-related injuries and saving many lives.

Fast and effective prediction of future paths of surrounding vehicles and consequently making automated adaptive decisions will improve the safety and efficiency of autonomous vehicles and driving assistance systems, especially in complex and less predictable scenarios caused by major contributors to accidents. Agile and accurate trajectory predictions will also improve the decision-making of autonomous vehicles towards enhancing ride comfort, energy consumption, and traffic congestion [27, 28]. In 2019, there were 36,096 fatalities on roadways in the United States [3, 4]. NHTSA (2019) also estimates that 11.9% of them involved a vehicle maneuvering in a manner that may be unpredictable to the other drivers (i.e., turning left or right, stopping or slowing in traffic, merging/changing lanes, or passing another vehicle). Such crashes at highway speeds, given the short TTC and limited distance range,

cannot be prevented with vision-based systems alone [29].

Real-time trajectory prediction on-spot is a quite challenging task due to the uncertain and dynamic nature of roadways. We often observe many non-linearities on vehicle trajectories, stream from nearby vehicles or the number of available lanes (environmental factors), or individual decisions or preferences (intrinsic factors). Predicting multiple possible trajectories for an active subject in the scene [16, 27] is a common practice. These trajectories are ranked based on the probability distribution of the prediction model, which makes them inherently less practical in real-time scenarios. Some recent approaches also consider the interactions of the nearby vehicles to successfully predict the path of a moving vehicle [15, 16]. However, they often have a relatively large model size and high computational complexity. A larger model size translates to higher storage requirements and higher system cost [30]. Smaller model size translates to faster performance and low memory requirement. Data storage and compression mechanisms are also crucial to reducing the system size and cost in V2X infrastructure.

This article proposes DeepTrack as a novel deep learning algorithm with comparable accuracy to best-in-class trajectory prediction algorithms but with a much smaller model size and lower computational complexity. DeepTrack encodes the vehicle dynamics with the aid of Temporal Convolutional Networks (TCN) instead of well-established mainstream Long-Short-Term Memory (LSTM) units. TCN, with its depthwise convolution as its backbone, can shrink the complexity of models and boost gradient flow for a more generalized trained model compared to LSTM-based solutions. We also augment DeepTrack with time attention modules to enhance the robustness against the noise and provide higher accuracy with minimum computational overhead. For the experimental results and comparison, this article uses datasets provided by the Federal Highway Administration (FHWA) under Next Generation Simulation (NGSIM) program.

Compared to CS-LSTM [27], DeepTrack reduces Average Displacement Error (ADE) by 12.23%, Final Displacement Error (FDE) by 2.69%, and also reduces the number of operations and model size by about 21.67% and 43.13%, respectively. CF-LSTM [31] is outperformed by DeepTrack by 2.43% in terms of ADE while operations and model sizes are reduced by 22.37% and 43.75%, respectively. The DeepTrack model reduces the number of operations and model size by 22.84% and 12.61%, respectively, over STA-LSTM [28], while ADE is lower by 5.97% (around maximum of 15 cm) and FDE is lower by 2.77%.

Overall, the key contributions of this article are:

- A lightweight trajectory forecasting model to precisely predict the location of vehicle of interest up to 5 sec. in the future.
- A novel encoder design based to reduce the complexity and size of the overall network by at least 22% and 12% compared to state-of-the-art trajectory prediction models with comparable accuracy.
- An extensive analysis on the design of the proposed network highlighting the effect of various components on the performance.

2.2 Related Work

Networks that predict vehicle trajectories can be broadly grouped into three categories: models based on physics, those that focus on maneuvers, and models that are interaction-aware [26]. While models rooted in physics adhere to physical laws, maneuver-centric models prioritize driver intentions. On the other hand, interaction-aware models emphasize the roles of surrounding vehicles and their interplay in forecasting motion and path. Historically, such interaction-focused models didn't gain much traction. Most older models either employed Dynamic Bayesian methods [32, 33] or were based on prototype trajectory designs [34]. Recent advances,

however, have seen a surge in models that leverage environmental context for better predictions. A significant number of these architectures incorporate Long Short-Term Memory (LSTM) neural networks [35, 28, 36] to assimilate details about nearby vehicles. Notably, in Deo *et al.*'s contribution [16], vehicle data is synthesized into a singular tensor, followed by convolutional and max-pooling stages to enhance adaptability. This data, combined with the LSTM encoding of the primary vehicle, feeds into a maneuver-focused LSTM decoder, elucidating multiple possible trajectories.

The study by Mercat *et al.* [37] adopts a similar LSTM-based structure but sidesteps predefined maneuver classifications. The model has intermediary self-attention layers that receive time-specific encoded details of each vehicle, ensuring a consistent input size regardless of vehicle count variations. This central segment formulates an attention matrix, leveraging features from the encoder, culminating in the decoder outlining path likelihoods for each vehicle. Notably, the LSTM-driven model in [37] outperforms many leading trajectory prediction algorithms. Nonetheless, Graph Neural Networks (GNN) have displayed superior accuracy in trajectory predictions in certain studies [31, 38].

Xie. *et al.* [31] introduced a GNN-based model [39] termed the teacher-student approach, which boasts enhanced accuracy in trajectory predictions compared to its predecessors. This model integrates a frame-wise graph reflecting agent positions, coupled with a Graph Convolutional Network encoder-decoder for generative learning and a Gaussian mixture model for congestion pattern creation. An LSTM-centric encoder-decoder in the student model refines trajectory predictions by aligning with the teacher model's congestion patterns.

Much of the contemporary literature underscores the progressive decrease in error rates for deep learning models in vehicle trajectory predictions. Our approach achieves commendable error rates, mirroring leading algorithms, but with a more streamlined model structure and reduced size. Our unique ATCN encoder captures vehicle posi-

tions over a recent three-second span, contrasting with contemporary LSTM encoders [31, 37]. After processing the encoder output, we utilize 2D convolution to distill past vehicle interactions. This data then informs our LSTM Trajectory predictor, forecasting a context-aware path for the target vehicle over the upcoming five seconds. Our model's training and validation leveraged the US Highway 101 [2] and Interstate 80 Freeway [1] datasets from the Federal Highway Administration's (FHWA) Next Generation Simulation (NGSIM) initiative.

2.3 Motivation: Applications of Real-time Trajectory Prediction in Highways

Predicting vehicle trajectories on the highway offers a variety of safety applications, particularly where points of conflict increase between road users. There are expected V2I applications to aid active traffic management systems to divert traffic away from predicted conflicts through lane-use control signals, dynamic hard shoulder running, and variable speed limits [40, 41]. Leveraging predictions to calm traffic through speed reductions or divert upstream traffic away from high conflict zones would be expected to reduce the potential for other vehicles to be impacted by potential collisions or constrain traffic flow. Moreover, such predictions may also reduce the likelihood of "secondary crashes" occurring in the aftermath of primary crashes [42], which are estimated to account for up to 15% of all freeway crashes [43]. In congested freeway platooning conditions, free-flowing traffic is at risk of propagating increasing braking responses upstream due to downstream hard braking, close-call, or collision events. The increased braking responses, paired with slow response times, propagated upstream risk rear-end collisions. Providing real-time traffic control to upstream traffic based on downstream traffic predictions of safety-critical events may offer the potential to dampen the effects of traffic instability by reducing time lags in response [44].

Further, there are expected V2X applications which may reduce the risk of predicted vehicle trajectories into high-risk areas such as work zones or roadside emergency work areas by providing advanced alerting to on-duty workers to seek safety [45].

Emergency responders are at risk of being struck by vehicles while conducting traffic stops or attending to a crash. Often these crashes occur due to distracted or impaired drivers unintentionally leaving their lane of travel [46]. Secondary crashes are also a risk for emergency responders when traffic is not properly controlled around and upstream from the scene [47]. Providing predictions of vehicle trajectories near the roadside, on-duty workers may also help to guide better traffic management through Portable Changeable Message Signs (PCMS) to alert drivers to risks in addition to emergency alerting to the workers themselves.

2.4 DeepTrack Design

This section discusses the DeepTrack architecture in detail.

The inputs of DeepTrack are defined as:

$$\boldsymbol{X}_{ego} = \begin{bmatrix} l_e^{t-h} & l_e^{t_1-h} & \cdots & l_e^{t_0} \end{bmatrix}, \tag{2.1}$$

$$\boldsymbol{X}_{nbr} = \begin{bmatrix} l_0^{t-h} & l_0^{t_1-h} & \cdots & l_0^{t_0} \\ \vdots & \vdots & \ddots & \vdots \\ l_j^{t-h} & l_j^{t_1-h} & \cdots & l_j^{t_0} \\ \vdots & \vdots & \ddots & \vdots \\ l_{N-1}^{t-h} & l_{N-1}^{t_1-h} & \cdots & l_{N-1}^{t_0} \end{bmatrix}, \tag{2.2}$$

where $l_i^t = \langle x_i^t, y_i^t \rangle$ is the position of vehicle i at time t, h is number of seconds of the past trajectory used for prediction, $\boldsymbol{X}_e$ is the car of interest, $\boldsymbol{X}_{nbr}$ is neighbour cars, and N is the number of vehicles. The shape of $\boldsymbol{X}_{nbr}$ is shown in Fig. 2.1. In a similar way, the output can be defined as follows:

$$\hat{\boldsymbol{Y}} = \left[l^{t_1}, l^{t_2}, \cdots, l^{t_f} \right]. \tag{2.3}$$

2.4.1 Model Architecture

We present the proposed DeepTrack architecture in Fig. 2.2. It consists of an encoder, a Vehicular Interactive Aware Convolution (VIAC), and an LSTM trajectory encoder. In the following sections, we explain the design and working of each component.

2.4.1.1 DeepTrack Encoder

Inspired by the generic TCN architecture, we propose an encoder to reduce model complexity and memory footprint. Fig. 2.1 shows the structure of a single DeepTrack encoder block. The DeepTrack encoders embed vehicles' path histories, both neighbors and target vehicles, into higher dimensions to capture their past trajectory. As opposed to previous works [31, 16], DeepTrack does not require dense layers to embed input features as needed for the LSTM encoder. Encoder convolutional operators capture and map the sequence of l^t by applying $K = [2, k]$ as a kernel, where $k \in \mathbb{N} | k \leq h$. As a result, DeepTrack has less model complexity, and better gradients flow from output to input during optimization.

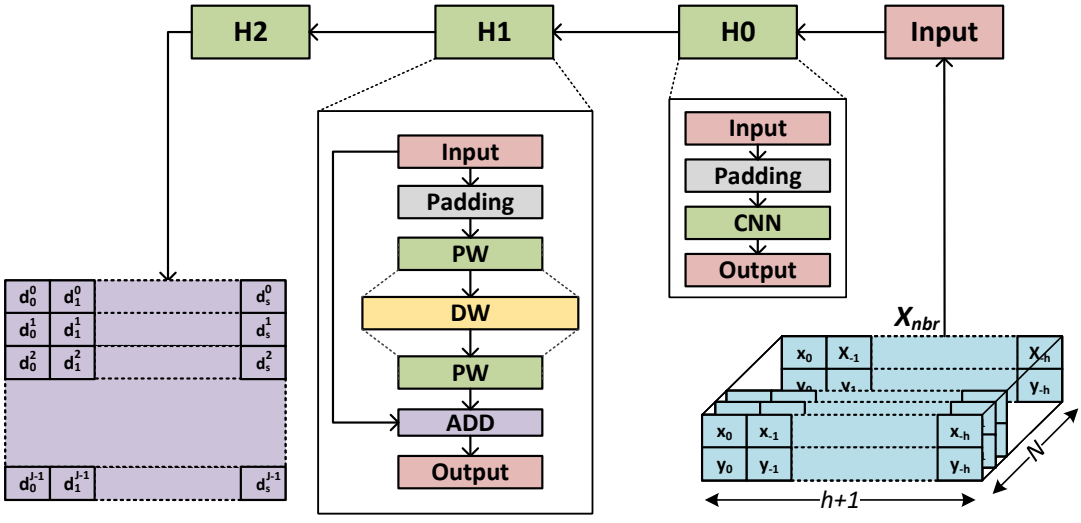

Figure 2.1: Structure of TCN based DeepTrack encoder.

2.4.2 Preliminaries and Problem Formulation

Traditional Convolutional Neural Networks (CNN) are used in computer vision applications due to their success in capturing spatial features within a two-dimensional frame. Recently, research has shown that specialized CNNs can recognize patterns in data history to predict future observations. This gives researchers interested in time-series forecasting options to choose other options over RNNs, an established DNN for time-series predictions. In one such case, TCN achieved the state of the art accuracy in sequence tasks, e.g., polyphonic music modeling, word and character-level language modeling, and audio synthesis [48, 49, 50].

TCNs are designed around two basic principles: 1) the convolutional operations are causal, i.e., predictions are made based only on current and past information; 2) the network receives an input sequence of arbitrary length and maps it to an output sequence of the same length. The use of causal convolutions in WaveNets [51] showed that it allowed for faster training as compared to LSTM based networks as they do not rely on recurrent connections. However, as the causal convolution needs large number of layers to increase the receptive field, WaveNet uses dilated convolutions to address this problem. In Dilated convolutions, the kernel is stretched to cover a larger part of the input. This is achieved by inserting holes (zeros) between the kernel elements. The level of enlargement is determined by dilation rate, which defines the number of spaces inserted between the kernel elements. Generally, d-1 spaces are inserted for dilation rate of d.

Simple causal convolutions have a dilation rate of 1, but other researchers incorporate dilated convolutions to scale the receptive field exponentially. The dilated convolution of F on element s of a sequence X is given as:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i}, \tag{2.4}$$

where $X \in \mathbb{R}^n$ is a 1-D input sequence, $*_d$ is dilated convolution operator, $f : \{0, ..., k-1\} \in \mathbb{R}$ is a kernel of size k and d is the dilation rate [51]. Also receptive filed of a dilated convolution can be calculated by:

$$rf = 1 + \sum_{j=1}^{L} [k(l) - 1] \times d(l), \tag{2.5}$$

where $j \in \{1, 2, 3, ..., L\}$ is the layers, k is the kernel size, and $d(j)$ is the dilation rate at layer j. This means that as the depth of the network increase, so does the receptive field. To address the issue of vanishing gradients resulting in exponentially expanding receptive fields with increasing network depth, TCN replaces standard convolutional layer in the residual block [52]. This is a widely used approach for convolutional architectures as it provides a path for information to pass through the layers. A residual block can be represented as:

$$y = \mathcal{F}_i(X) + X \tag{2.6}$$

where, y is the output of residual block, and $\mathcal{F}_i$ represents the operations such as, convolutions layers , non-linearity, and normalisation applied to input X at each layer i. The residual block helps the network in learning the modifications applied to input at each layer [51].

DeepTrack uses two different encoders. There is one shared by all neighbors, shown by the shaded box in Fig. 2.2, which maps their dynamics to higher dimensions so that the VIAC can comprehend their interdependencies. The other encoder maps only the ego dynamics, as illustrated by the red box in Fig. 2.2.

Since we have a padding unit in each encoder to make sure the input and output of standard and depthwise convolution will be the same (second principle of TCN

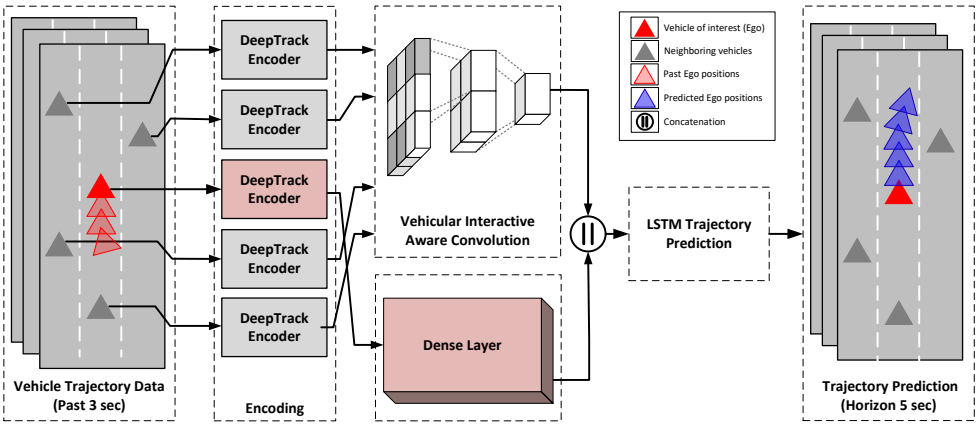

Figure 2.2: The path prediction model workflow. The location of the neighbors (gray triangles) and car of interest (solid red triangle) is shown at t_0 in Vehicle trajectory data (extreme left) block and Trajectory prediction (extreme right) block. Triangles denoting semi-transparent red in Vehicle trajectory data block, and semi-transparent blue in Trajectory prediction block represent observed history paths, and model output respectively. The observed history paths of neighbours (for past 3 seconds) are used by the model but not shown in the figure to avoid confusion.

architectures), the $2p$ zeros are added symmetrically, where p is given by:

$$p = \lceil \frac{(o-1) \times s + (k-1) \times (d-1) - i + k}{2} \rceil, \tag{2.7}$$

where o is the output size, i is the input size, s is the stride, k is the kernel, and d is the dilation. According to Eq. 2.7, if we increase the kernel size or dilation, more zeros should be padded to the input. The addition of excessive zeros to the input has two main disadvantages: ① it degrades the model's performance due to redundant zeros, and ② it increases the model computational complexity. As a result, we set the dilation and kernel size of the DeepTrack encoders for different models as shown in Table 2.2, both DeepTrack encoders have three hidden layers; however, the output dimensions differ.

Each DeepTrack encoder has a three-layered structure with a padding block to ensure that the input and output sizes are the same, followed by a convolutional block. Layer H0 uses standard convolution ($d = 1$). As a widely accepted practice in the deep learning community, batch normalization [53] layer is added after each convolution to

speed up and stabilize the model training followed by a ReLU activation. We have intentionally not shown Batch Normalization (BN) and ReLU activation in Fig. 2.1 for simplification of the diagram.

For DeepTrack, Swish, as well as ReLU activation, were tested after BN. Swish is very similar to ReLU but does not abruptly change its direction. However, we use ReLU activation for DeepTrack to provide marginally better results. BN and ReLU are not shown in Fig. 2.1 for simplification.

In the subsequent layers (H1-H2), a padding block is followed by a pointwise (PW), depthwise (DW), and PW convolutions instead of a CNN to reduce the model complexity. As Layers H1 and H2 are identical, details of H2 are not shown in Fig 2.1. BN and ReLU activation is applied to the output of the last PW convolution before adding it to the input of the residual block [52] as expressed by Eq. 2.6.

To the best of our knowledge, we are first to present a deep learning algorithm with a modified generic TCN architecture for trajectory prediction.

2.4.2.1 Attention Mechanism

Attention mechanisms have been used to better interpret the model and extract the significant information. It has also been used in multiple trajectory prediction applications as shown in [54, 55].

In this architecture, additive attention mechanism [56] for both DeepTrack encoders to guide decoders are used to predict the trajectory based on the importance of features, similar to the work by Lin *et al.* [28]. In order to get the importance of the encoded output, the first associated weight score vector, $\overrightarrow{\omega}$, should be calculated by:

$$\overrightarrow{\omega} = tanh(WD), \tag{2.8}$$

$$\boldsymbol{D} = \begin{bmatrix} l_0^{t_0} & l_0^{t_1} & \cdots & l_0^{t_s} \\ \vdots & \vdots & \ddots & \vdots \\ l_{J-1}^{t_0} & l_{J-1}^{t_1} & \cdots & l_{J-1}^{t_s} \end{bmatrix}, \tag{2.9}$$

where D is the DeepTrack encoder output, s is the encoder output length size set to h, J is the last output channel size, and W is trainable parameter. The final attention score is then given by:

$$f_{att} = \sigma(\overrightarrow{\omega}), \tag{2.10}$$

$$D_{final} = D^T f_{att}, \tag{2.11}$$

where σ is $Softmax$ function. Fig. 2.3 illustrates the attention mechanism. The output of $softmax$, f_{att} is the importance heatmap, and it will be again multiplied with the encoded data. Based on the focused data, the VIAC and LSTM trajectory decoder can figure out the vehicle interactions and generate the final prediction effectively.

2.4.2.2 VIAC

Analyzing the interaction between the ego and its neighbors is necessary to predict the future trajectory for the vehicle of interest. Despite capturing individual behavior, the DeepTrack encoder is unable to comprehend the entire scene. Social pooling proposes a solution by pooling encoded data around a specific target [57]. The task is accomplished by defining a spatially correlated grid $f \times g \times N$ regarding the car of interest. Similar to Social Pooling [16], we set f and g to 13 and 3, respectively. The structure of the 3×3 grid that masks the encoder output is shown in the VIAC

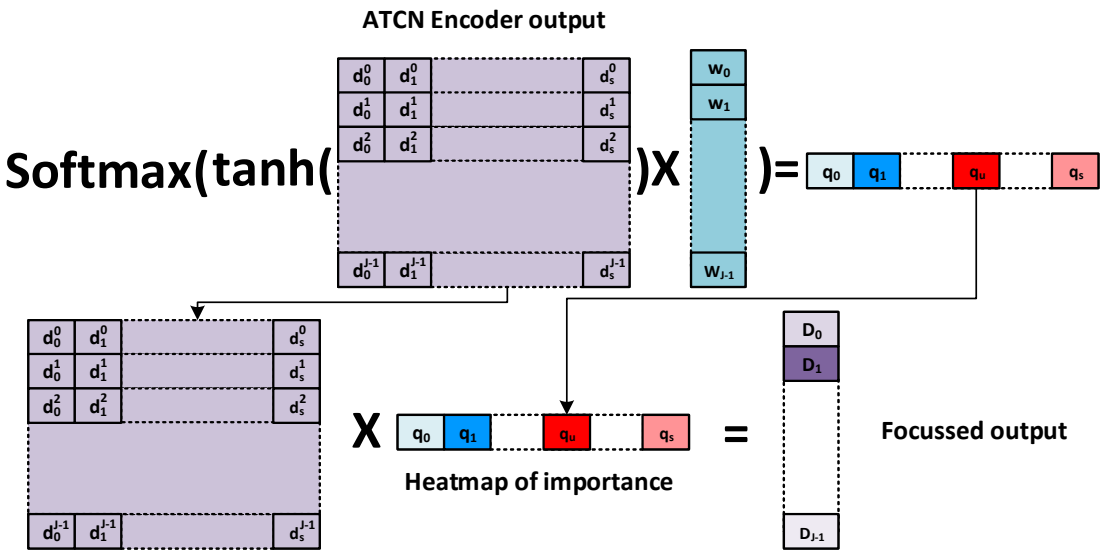

Figure 2.3: Additive Attention. The output of softmax is multiplied with DeepTrack encoder output to lead the next layers on essential features.

block of Fig. 2.2.

DeepTrack comprehends the interdependencies of the vehicle by applying convolutions to embed information. The neighborhood dynamics are encoded and mapped to the lower dimension using the two-layer convolution and a pooling unit. The convolutional layers help extract the local features from the spatial grid around the ego vehicle. The use of convolutional-social pooling in the VIAC shows lower performance degradation as compared to fully connected social pooling, as shown by Deo *et. al* in [16].

The additional dense layer dedicated to the dynamic encoding of the ego vehicle is fully connected. The dense layer also remaps the DeepTrack encoder output of ego to have the same feature size so that it can be concatenated with the result of VIAC as shown in Fig. 2.2. The concatenated output is then passed through the LSTM Trajectory Prediction block.

2.4.2.3 LSTM Trajectory Prediction

Only at the final stage, DeepTrack uses an LSTM-based encoder to predict the future trajectory, $\hat{Y}$. We have not used TCN based encoder as the final stage because TCN can map the temporal information only to a higher output channel. Similar to what is accomplished by the DeepTrack encoder at the first stage. LSTM is only used to map and decode the VIAC and Dense layer's concatenated output to the final output prediction.

2.4.3 Algorithm

Algorithm 1 represents a step-by-step working of the DeepTrack prediction network. Input X_{nbr} has dimension $(N-1) \times (h+1)$, and X_{ego} has dimension $1 \times (h+1)$. Expected output, $\hat{Y}$, is given by eq. 2.3.

2.4.3.1 Encoder Functions

There are two encoder functions in Deeptrack, $Encoder_{nbr}$ and $Encoder_{ego}$. Both Encoder functions have three hidden layers represented by blocks H0, H1, and H2 in Fig. 2.1. H0 uses standard convolution, but H1 and H2 use point-wise, depth-wise, point-wise convolutions. All the convolution operations are followed by appropriate padding, normalization, and activation. Number feature sizes for each block are as shown in Table 2.2.

First, $Encoder_{nbr}$ and $Encoder_{ego}$ functions is applied to input vectors X_{nbr} and X_{ego} respectively. Next, to incorporate the attention mechanism, Softmax function is applied to tanh activation of the dot product of $Encoder$ outputs D_{nbr}, and D_{ego} with W_{nbr}, and W_{ego} respectively. W represents trainable attention weight vectors, and σ represents $Softmax$ function. The attention mechanism is a part of encoder blocks in Fig. 2.2. The output of attention block is then passed to VIAC.

2.4.3.2 VIAC, Dense Layer and $LSTM_{decoder}$ Functions

VIAC and Dense Layer (DS) functions are applied to the dot product of the transposed output of *Encoder* functions and attention mechanism for neighbor and ego vehicles. VIAC first combines the tensors for every grid with a car to form a single tensor. Next, a couple of convolutional layers are applied to this tensor, followed by a pooling layer. Simultaneously, the decoded state tensor of the ego vehicle is passed through a fully connected layer represented by DS [16]. DS translates the inputs into a feature size that is concatenated with VIAC output to produce a comprehensive encoded trajectory, Y_{final}. Concatenation is represented by '||' in the algorithm.

Finally, $LSTM_{decoder}$ function is applied to Y_{final}. The decoder has two softmax layers with outputs concatenated to predict the final trajectory of the ego vehicle, $\hat{Y}$.

Algorithm 1 DeepTrack prediction model pseudo code

Input Neighbour: X_{nbr} = all rows except row l from metrics in eq. 2.2

Input Ego: X_{ego} = row l from metrics in eq. 2.1.

Output:$\hat{Y}$ = output as shown in eq. 2.3 Initialize encoder layers, kernels, dilation

$D_{nbr} = Encoder_{nbr}(X_{nbr})$

$f_{att\_nbr} = \sigma(tanh(W_{nbr}D_{nbr}))$

$y_{nbr} = VIAC(D_{nbr}^{T} f_{att\_nbr})$

$D_{ego} = Encoder_{ego}(X_{ego})$

$f_{att\_ego} = \sigma(tanh(W_{ego}D_{ego}))$

$y_{ego} = DS(D_{ego}^{T} f_{att\_ego})$

$y_{final} = y_{ego} \;||\; y_{nbr}$

$\hat{Y} = LSTM_{Decoder}(y_{final})$

2.5 Evaluation

The performance of DeepTrack is evaluated using NGSIM's widely used I-80 [1] and US-101 [2] vehicle trajectory datasets. A rate of 10 Hz is used for sampling

the vehicle's trajectory for 45 minutes. Each dataset includes three segments of 15 minutes long of mild, moderate, and congested traffic. Similarly to [31, 37, 28], we divided the dataset into three parts: training, validation, and testing. The dataset provides around 8 million data entries divided into 70% training data, 10% validation data, and 20% test data. As discussed in CS-LSTM [16], DeepTrack also uses a stationary frame of reference.

The direction of motion of the vehicles is defined by the head of the triangle as shown in Fig. 2.2. Lanes immediately next to the ego vehicle are considered for tracking the neighbors' position. This helps capture the effect of movement of immediate neighbors on the ego vehicle as they have maximum influence on its future trajectory. The area around the ego vehicle is converted into a 13×3 size grid, with each grid cell 15 feet long and width equal to lane width. The ego vehicle is assigned the center cell, whereas each neighboring vehicle in the 13×3 grid is assigned a cell based on the position of its front bumper around the ego.

Based on the work of [16], each trajectory is also segmented into 8 seconds, where the first three seconds are used as a path that was observed, and the model will predict the following five seconds. Previous works downsampled each second by two [31, 37] to reduce the complexity of the LSTM encoder. While we are not limited to this fact, we also downsampled the inputs for a fair comparison. In the following subsections, we discuss the DeepTrack implementation environment, the effect of various components

Table 2.1: DeepTrack Variants (Transposed)

	Features	**Convolution**	**Activation**	**Optimizer**
DT_0^*	[16, 32, 64]	Separated	Swish	ADAM
DT_1	[32, 16, 64]	Separated	ReLU6	ADAM
DT_2	[32, 16, 64]	Separated	Swish	ADAM
DT_3	[32, 16, 64]	Normal	Swish	SGD
DT_4	[32, 16, 64]	Normal	Swish	ADAM
DT_5	[16, 32, 64]	Normal	Swish	ADAM
DT_6	[16, 32, 64]	Normal	ReLU6	ADAM

on the design, quantitative results by comparing with contemporary models, and qualitative prediction analysis for various scenarios. All models proposed in this paper are implemented using the PyTorch package, an open-source machine learning library. The training was performed on Nvidia Tesla V100 GPU for 30 epochs using the ADAM optimizer with a default learning rate of 0.001.

2.5.1 Evaluation Metrics

A comparison of DeepTrack against off-the-shelf algorithms on the NGSIM dataset was conducted to provide a comprehensive comparison. Root Mean Square Error (RMSE), Average Displacement Error (ADE), and, Final Displacement Error (FDE) are used as a measure of prediction accuracy and performance of the system. As DeepTrack is designed with edge-based real-time applications in mind, the number of MACs and parameters of the models are also compared to state-of-the-art models to present a perspective on model complexities. The RMSE at time t is given by:

$$RMSE^t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Y_i^t - \hat{Y}_i^t\right)^2}, \tag{2.12}$$

where Y is the ground truth, $\hat{Y}$ is predicted output, and N is the number of samples. Average Displacement Error (ADE) and Final Displacement Error (FDE) are also calculated to compare the average RMSE over 5 seconds and error in the final predicted position.

ADE refers to the mean square error (MSE) overall estimated points of every trajectory and the actual points, and FDE is the root mean square error distance between the final predicted trajectory points and ground truth. These evaluation parameters are used in the following sections to compare and analyze the performance of DeepTrack.

2.5.2 DeepTrack Models and Comparisons

This section helps in understanding the impact of various components on the performance of DeepTrack. Different variants of DeepTrack were designed and evaluated to analyze the influence of each alteration on the network's overall performance.

2.5.2.1 Variant Models Design

The variant models were designed by changing or removing one of the following components from the network: attention-mechanism, optimizer, activation function, convolution type, and neighbors output feature sizes. The attention mechanism helps in grasping the effect of the neighboring vehicles on the ego trajectory. A DeepTrack model without an attention mechanism was designed, and the results are discussed here. In this study, ADAM and Stochastic gradient descent (SGD) optimizers were utilized to adjust model parameters to reduce the training loss. Swish and Rectified Linear Unit (ReLU) were used as an activation function for all the layers in DeepTrack analysis. Separated and normal convolution-based networks were designed and tested to reduce the complexity and compare performances of the network. Table **??** shows seven different variants of DeepTrack each varying from other in at-least one of the above mentioned aspects. The model with superscript *, e.g., DT_0^*, denotes an absence of the attention-mechanism.

Table 2.2: ATCN encoder configuration for ego vehicle and neighbors.

ATCN Encoder	**Configurations (H0, H1, H2)**		
	Output feature dimensions	**Dilation rate**	**Kernel size**
Neigbours	[16, 32, 64]	[1, 1, 1]	[2, 2, 2]
Ego	[8, 16, 32]	[1, 1, 1]	[2, 2, 2]

As a part of the ablation study, several different designs of TCN-based trajectory prediction networks were studied. However, only seven models based on their effect on the overall system's performance are presented in this study. First three models DT_0^*, DT_1, and DT_2 use separated convolution with a combination of different activation

functions, Attention-mechanism layer, and neighbour output features. Next four models DT_3 DT_4, DT_5, and DT_6 used normal convolution with a combination of other parameters. All the models were trained with a data split similar to [16]. The hidden layers, dilation rate and kernel sizes of various model encoders are shown in Table 2.2, DT_i represents models DT_1 to DT_6 as the encoder parameters are fixed models with attention mechanism to limit excessive padding.

2.5.2.2 Performance Comparison and Model Complexities

Table 2.3 shows the performance of DeepTrack variants in three areas, mean error at the end of each second in meters (RMSE) , final and average displacement errors calculated in meters (FDE, ADE), and model complexity (number of MACs and Parameters).

DT_2 and DT_5 have the best performance in terms of RMSE. DT_2, DT_5 and DT_6 have best ADE values, while DT_5 and DT_6 have the best performance in terms of FDE. There is only a 1.25% (0.04 m) difference in the FDE of DT_2 and the best performing models. Models DT_1 and DT_2 prove to be best in complexity-based performance, which is one of the most important aspects of this study. Hence, it can be concluded that DT_2 with attention mechanism, ADAM optimizer, ReLU activation, separated convolutions, and output features of [32, 16,64] is has the best overall performance

Table 2.3: Performance comparison of DeepTrack variants based on Root mean square error, Displacement errors, and Computation.

Model	**RMSE (m)**					**FDE**	**ADE**	**Complexity**	
	1s	**2s**	**3s**	**4s**	**5s**			**MACs**	**Params**
DT_0^*	0.45	1.13	1.90	2.84	4.03	3.34	2.07	2.91M	171K
DT_1	0.46	**1.07**	1.84	2.78	3.93	3.23	2.02	**2.80M**	**109K**
DT_2	0.47	1.08	**1.83**	**2.75**	**3.89**	3.25	**2.01**	**2.80M**	**109K**
DT_3	**0.44**	1.14	1.92	2.86	4.01	3.27	2.07	3.22M	125K
DT_4	0.46	1.12	1.89	2.82	3.96	3.24	2.05	3.03M	118K
DT_5	0.46	1.08	1.84	2.76	3.90	**3.21**	**2.01**	4.14M	125K
DT_6	0.45	1.09	1.85	2.77	3.9	**3.21**	**2.01**	3.22M	125K

among all the DeepTrack variants considered in this study.

The model with SGD optimizer, Swish activation, and standard convolution, DT_3, and one without attention mechanism, DT_0^*, have the worst RMSE except for the first second. DT_0^* has the worst FDE and ADE performance, making a solid case for the use of ADAM optimizer. DT_5 has the worst performance in terms of complexity with 32.4% higher MAC count than DT_2 and a joint highest in number of Parameters with DT_6. Other models have similar performances in terms of RMSE, FDE, and ADE, but the difference can be observed when the complexity of the algorithms is analyzed. As expected, the models using standard convolutions show higher complexity when the number of multiply-and-accumulates (MACs) and model parameters are compared. The number of MACs and parameters are lowest for DT_1 and DT_2 as they use separated convolutions resulting in lower complexity than all the models using standard convolution.

It can be concluded that using separated convolutions helps in reducing the complexity of a model. Comparison of models DT_1 and DT_4 shows that the introduction of separated convolution helps reduce the number of MACs by 7.5% and the number of parameters by 8.1%. The attention mechanism also helps improve overall performance, as shown in the comparison of DT_0^* and DT_1. The use of ADAM optimizer is also justified by analyzing the mediocre performance of DT_3 with SGD optimizer. The analysis of effect of various factors on performance of DeepTrack continues as we present the effect of different amount of training data on the model performance in the next part.

2.5.2.3 Generalization Study

Table 2.4 summarises the error-based performance of DT_2 as it is the best performing DeepTrack variant. Column 1 in table 2.4 shows the data split ratios as Tr:Val:Ts representing train : validation : test set ratios respectively. The data-split of 70:10:20 is same as used in [16] for fair comparison with other models discussed in next section.

The results of data split of 80:10:10 as compared to 70:10:20 shows a possibility of over-fitting as the error is higher for larger training dataset. Other datasets with 60% and 50% training data-split also show an increase in root mean and displacement errors for DT_2.

Table 2.4: Performance comparison of best DeepTrack variant DT_2 based on different data splits of NGSIM dataset. RMSE, FDE, ADE are in meters.

Dataset split	**RMSE horizon**					**FDE**	**ADE**
Tr:Val:Ts	**1s**	**2s**	**3s**	**4s**	**5s**		
80:10:10	**0.46**	1.1	1.9	2.88	4.09	3.37	2.06
70:10:20	0.47	**1.08**	**1.83**	**2.75**	**3.89**	**3.25**	**2.01**
60:25:15	0.5	1.2	1.98	2.92	4.07	3.34	2.13
50:30:20	0.56	1.25	2.06	3.03	4.22	3.47	2.22

2.5.3 Comparison Against Existing Approaches

We compare the results of DeepTrack against the four prominent recently introduced models. (1) Convolutional-social-LSTM (CS-LSTM) [16]: It is based on Social-LSTM [57], an algorithm used for human trajectory detection. CS-LSTM is an encoder-decoder-based model using a social pooling layer to extract the features from the interaction of vehicles in every input sample. (2) CF-LSTM [31]: A student-teacher network introduced for trajectory prediction. In this network, the LSTM Encoder-Decoder-based model is used for student algorithm and the convolutional graph network for teacher algorithm. (3) Spatiotemporal attention-LSTM

Table 2.5: Prediction and Model complexity comparison of DeepTrack with Trajectory forecasting models based on NGSIM dataset.

Models	**RMSE (m)**					**FDE**	**ADE**	**Complex.**
	1s	**2s**	**3s**	**4s**	**5s**			**MACs/Params**
CS-LSTM [16]	0.61	1.27	2.09	3.10	4.37	3.34	2.29	3.58M/191K
CF-LSTM [31]	0.55	1.1	1.78	2.73	3.82	-	2.06	3.61M/193K
SAAMP [37]	0.51	1.13	1.88	2.81	3.98	-	2.06	-/-
STA-LSTM [28]	**0.37**	**0.98**	**1.71**	**2.63**	**3.78**	3.16	1.89	3.63M/124K
(DT_2)	0.47	1.08	1.83	2.75	3.89	3.25	2.01	**2.80M/109K**

(STA-LSTM) [28]: As the name suggests, STA-LSTM uses spatial and temporal information with an attention mechanism to explain the effect of historical trajectories and neighboring vehicles on the ego vehicle. (4) Social Attention Multi-Modal Prediction (SAAMP) [37]: This model used an LSTM-based encoder-decoder structure with attention layers in the middle to incorporate real-time interactions. It utilizes a multi-head attention mechanism and fuses the long-range attention for joint and multi-modal forecasts.

The performance of DeepTrack and all the models mentioned in section 2.5.3 are listed in Table 2.5. We compare the error and complexity of DeepTrack to state-of-the-art algorithms in vehicle trajectory prediction using NGSIM datasets. As DeepTrack aims to provide best-in-class trajectory forecasting with a low error rate, the following sections analyze the performance in terms of RMSE up to five seconds, FDE, ADE, and complexity of the DeepTrack with other networks.

2.5.3.1 Error-Based Analysis

Compared to CS-LSTM, CF-LSTM, and, SAAMP, DeepTrack can reduces ADE by 12.23%, 2.43% and 1.47% respectively as shown in Table 2.5. DeepTrack also excels for all the steps of RMSE comparison to CS-LSTM and SAAMP. The better performance of DeepTrack is since it has higher gradient stability due to the use of a TCN-based encoder that it is better able to generalize solutions. However, CF-LSTM is better than DeepTrack at 3rd, 4th, and 5th second, and STA-LSTM outperforms DeepTrack. STA-LSTM gives 5.97% and 2.77% better ADE and FDE performance than DeepTrack. Thus, DeepTrack under-performs when compared with STA-LSTM with a small margin.

2.5.3.2 Model Complexity Analysis

DeepTrack outperforms every algorithm in terms of the number of MACs and Parameters, as shown in Table 2.5. Analyzing and comparing the MAC operations

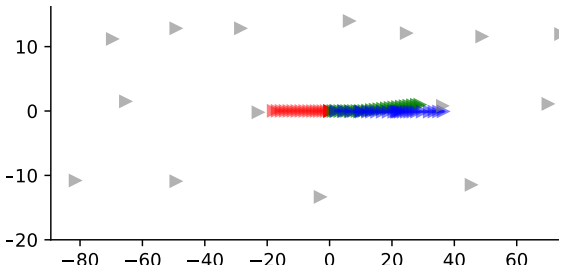

Figure 2.4: Congested traffic

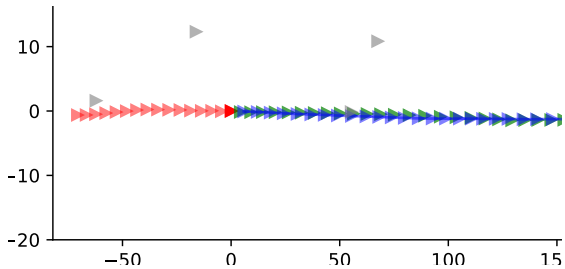

Figure 2.5: Lane-keeping

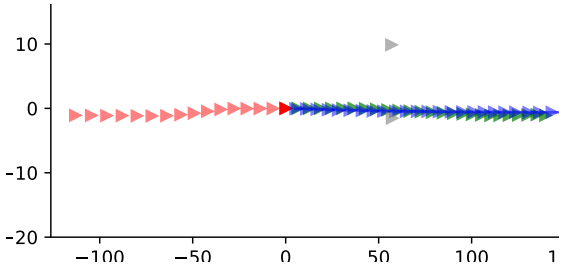

Figure 2.6: Maneuvering - Passing from left

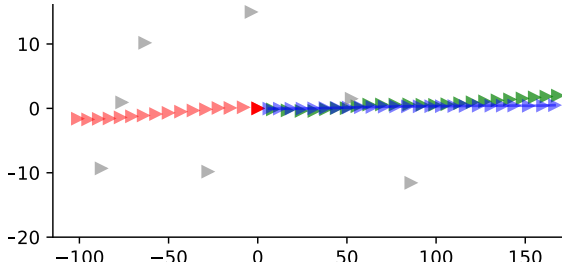

Figure 2.7: Maneuvering - Passing from right

Figure 2.8: Congested traffic scenario: the location of the neighbors (gray triangles) is shown at t_0. Triangles denoting red, green, and blue respectively, represent observed history paths, ground truth, and model output.

and the size of the model parameters for the approaches mentioned in section 2.5.3, we anticipated the difference. Deeptrack undercuts STA-LSTM in terms of complexity by 22.84% fewer MACs count and 12.61% better parameter count. It also provides 21.67% better MACs performance, 43.13% fewer parameters than CS-LSTM, and outperforms CF-LSTM by 22.37% and 43.75% in terms of the number of MACs and parameters, respectively. We could not compare its model complexity with DeepTrack and DeepTrack-ATT with SAAMP as the source code was not available publicly.

2.5.4 Qualitative Results

The analysis of DeepTrack output for different scenarios are discussed in this section. In Fig. 2.8-2.11, the location of the neighbors (gray triangles) are shown at t_0. Triangles denoting red, green, and blue represent observed history paths, ground truth, and model output. The model predicted output for four scenarios is shown as an aid to understanding the model behaviour: ① congested traffic (Fig. 2.4), ②

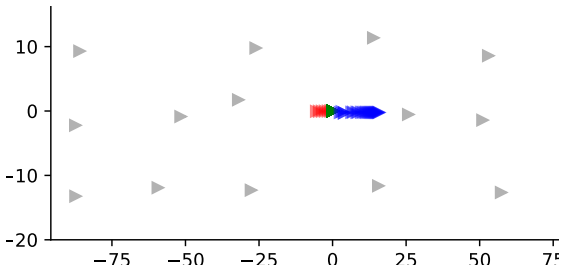

Figure 2.9: Congested traffic

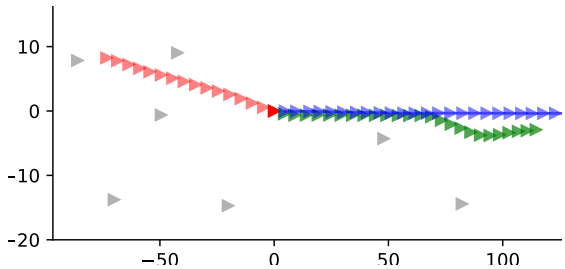

Figure 2.10: Multiple lane changing

Figure 2.11: Two instances where the model could not accurately predict the trajectory due to erratic driver behavior. The legend remains consistent with Fig. 2.8

lane-keeping (Fig. 2.5), ③ maneuvering and passing a car from left lane (Fig. 2.6), ③ maneuvering and passing a car from right lane (Fig. 2.7), and ④ cases where the model failed to predict the trajectory precisely (Fig. 2.11). The three types of the path shown in red, green, and blue triangles represent path history, ground truth, and predicted trajectory for the designated vehicle. The location of the neighbors (gray triangles) is also shown at t_0. For the sake of simplicity, we did not show the neighbors history path.

The comparison of Fig. 2.4 and Fig. 2.5 shows that the model can accurately estimate the velocity of the interest car based on the ego history. The model correctly predicted that vehicles would travel less distance as a result of congested traffic. The vehicle in the lane-keeping scenario travels farther, and DeepTrack has interfered with the same behavior. Figures 2.6 and 2.6 illustrate how DeepTrack performs when a car of interest passes its front vehicle from either the left or the right lane. Fig. 2.11 shows the scenarios in which DeepTrack was not able to predict the trajectories due to uncertainty in driver behavior. In the congested scenario (Fig. 2.9), although the driver slowly drove his car until t_0, the vehicle stopped for the entire next five seconds, while the model predicts it would come close to the front car.

2.6 Conclusion and Future Works

DeepTrack is a deep learning model with comparable accuracy to best-in-class trajectory prediction algorithms but with a smaller model size and lower computational complexity. The vehicle dynamics are encoded using a TCN-based encoder instead of LSTM units in DeepTrack, and TCN utilizes depthwise convolution, thereby reducing the complexity of models in terms of size and operations compared with LSTMs. The results indicate that DeepTrack reduces the model size and complexity by at least 21.67%, and 43.13% compared to CS-LSTM, 22.37%, and 43.75% compared to CF-LSTM, and 22.84%, and 12.61% than STA-LSTM. The RMSE and displacement errors for DeepTrack are better or comparable to most state-of-the-art trajectory prediction algorithms using NGSIM dataset used in this manuscript.

Acknowledgment

This work was supported by the National Science Foundation (NSF) under Award No. 1932524.

CHAPTER 3: A POV-BASED HIGHWAY VEHICLE TRAJECTORY DATASET AND PREDICTION ARCHITECTURE

3.1 Introduction

Intelligent Transportation Systems (ITS) have become integral to modern transportation networks, leveraging various technologies to improve road safety and efficiency. One key aspect of ITS is using trajectory prediction networks, which can accurately forecast the movements of vehicles, pedestrians, and other road users [11, 12, 13, 14]. In recent years, advancements in technology, such as accident detection systems [18, 19], traffic surveillance systems [58], and lane departure warning systems [59] have shown great potential in reducing accidents and saving lives on the road. However, highway safety has historically received less attention and innovation than other areas, possibly due to high costs and regulatory hurdles.

The National Highway Traffic Safety Administration reported 42,000 fatalities in 2022 [**?**] due to highway-related motor accidents. In addition, 857 fatalities were reported in 2020 [60] in work zone accidents, with a 4% increase in the death toll over the past two years [61]. These statistics demonstrate a significant scope for improvement in ITS applications related to highway surveillance and safety.

Vehicle trajectory datasets are essential for studying traffic behavior and analyzing macroscopic traffic data[18, 20]. However, most available datasets provide only bird's-eye or dashcam views [1, 2, 6, 7, 8, 9] or low-quality high-angle view videos[21, 17]. These perspectives may be insufficient for highway safety and surveillance applications[62], mainly when the environment is subject to variation. Eye-level and high-angle views of highway traffic with merging lanes are particularly important for highway-based safety applications, such as work zone safety[63], collision

avoidance[?] and lane departure warning systems[59], as well as surveillance applications, such as traffic monitoring and incident detection[18]. However, they are not typically found in trajectory datasets. High-angle view data, for example, can be used for incident detection and response, traffic monitoring[64, 17] and analysis[17, 65, 66], which can be challenging to obtain in real-time with bird's-eye view data. Similarly, Eye-level view data can be helpful for real-time pedestrian and worker safety and alarm applications[67, ?, 63] for work zones and highway management.

While many trajectory datasets have served as crucial benchmarks in their respective focus areas, they often lack multiple POVs of incoming traffic. They rarely examine the vehicle dynamics during lane merges which is essential in comprehending the naturalistic driving maneuvers close to the real-world worksite. This leaves a blind spot for research on highway-based edge ITS applications involving transportation safety and AI.

In light of this, we propose the *Carolinas Vehicle Dataset (CHD)*, a comprehensive trajectory dataset capturing naturalistic highway driving behavior from multi-POVs. CHD consists of 338,000 unique trajectories with five highway vehicle categories, varying geometries, and lane mergers. It also provides 1.6 million high-resolution frames with high traffic density recorded during varying lighting and weather conditions. Thus, CHD includes annotations and trajectory information for highway vehicles, making it a valuable resource for ITS and computer vision applications to advance the development of safer and more efficient transportation systems.

We also introduce *PishguVe*, a trajectory prediction architecture that leverages attention for graph isomorphism and convolution neural networks to achieve SotA performance across datasets with multiple POVs. PishguVe provides a solution to overcome the limitations of POV-based trajectory prediction approaches, enabling more generalizable models for real-world applications. Compared to existing approaches evaluated on widely used NGSIM dataset, PisguVe surpasses current SotA

Table 3.1: Comparison of existing trajectory datasets with CHD. T and BB in the Annotation column stand for Trajectories and Bounding Boxes (BB) respectively. FPS stands for Frames Per Second. BE stands for Bird's-eye-view, DC stands for Dashcam view, HA and EL stand for High-angle and Eye-level view respectively.

Dataset	Length (h)	Frames	BB	FPS	Lanes	Areas Covered	Annot.	POV
NGSIM	1.5	11.2K	-	10	Yes	2	T	BE
HighD	16.5	-	-	25	Yes	6	T	BE
Agroverse2 Fore.	763	-	-	10	No	2	T	DC
AppolloScape	100	93K	81.8K	10	No	4	T	DC
Lyft	118	232K	1.3M	10	No	1	T	DC
WAYMO	5.5	1M	12M	10	No	3	3D BB, T	DC
nuScenes	5.5	1.4M	1.4M	2	No	2	3D BB, T	DC
CHD (Ours)	7.5	1.6M	1.39M*	5, 60	No	4	2D BB, T	HA, EL

*Bounding boxes in CHD's trajectory data.

[68] with 12.50% and 10.20% improvement over ADE and FDE, respectively. It also outperforms existing approaches in ADE and FDE by 14.58% and 27.38% when evaluated on CHD eye-level and by 8.3% and 6.9%, respectively, on CHD high-angle POV dataset.

In summary, the contributions of this article are:

- We introduce the *CHD*, a vehicle trajectory dataset from multiple POVs capturing vehicle maneuvers on highways with varying geometries and lane mergers. CHD also provides 1.6 million high-resolution images with annotation data for detection and tracking applications.

- We introduce *PishguVe*, a SotA trajectory prediction network based on attention-based convolutional and graph isomorphism networks. PishguVe achieves a

best-in-class error rate when evaluated on multiple datasets, making it well-suited for real-time trajectory prediction in different POV scenarios.

- To verify the effectiveness of *CHD* and *PishguVe*, we present a comprehensive comparison of the trajectory dataset and assessment of trajectory prediction algorithms trained and evaluated on NGSIM and CHD in this manuscript.

3.2 Related Works

3.2.1 Vehicle Tracking and Trajectory Datasets

NGSIM dataset [1, 2] is one of the most extensive available datasets recorded in a bird's-eye view angle. This dataset is collected from two freeways, namely I-80 and US-101. This dataset provides trajectories extracted at a rate of 10Hz. HighD dataset [6] is recorded in six locations on German highways using drones at the bird's-eye POV. However, the original videos are not provided; only the extracted trajectories using computer vision algorithms are available. Agroverse 2 Motion Forecasting dataset [7] provides trajectory data for different classes of vehicles and pedestrians extracted from videos recorded in six cities. All the mentioned datasets are collected from a bird's-eye POV. However, most of the time, bird's-eye view videos are unavailable in real-world scenarios since traffic cameras are usually mounted on top of the traffic lights. Another group of datasets, such as ApolloScape Trajectory [69], Lyft [8], WAYMO [9], and nuScenes [10] are specifically designed for autonomous vehicle applications and are recorded from a dashcam POV. These datasets are useful as benchmarks for trajectory prediction for several applications. However, their applicability to real-world highway safety applications is limited due to the properties of the data, such as moving cameras and viewpoints. Table 3.1 shows a detailed dataset analysis.

For vehicle detection/tracking, some existing datasets with high-angle POV are listed in 3.2. The TRANCOS [70] dataset is recorded at the high-angle POV and is

a vehicle counting dataset captured using publicly available surveillance cameras in Spain. Unlike bird's-eye POV datasets, in TRANCOS, vehicles are highly overlapping and close to real-world scenarios. TRAF object tracking dataset [71] provides videos in high-angle and dashcam POVs. The UA-DETRA dataset [72, 73, 74] is a challenging real-world dataset captured from a high-angle point of view. It consists of videos captured from 24 locations in China and is manually annotated. Although the dataset is introduced for multi-object tracking and detection, it can also be used as a benchmark for trajectory prediction.

Furthermore, a set of datasets mainly focused on accident and traffic behaviors. TCP [75] introduces a high-angle POV dataset, capturing a four-way intersection at different times of the day. It uses an object detector for labeling the vehicles, and it has hand labels for when the vehicle enters the intersection. CADP [21] is another interesting dataset on car accidents. This dataset is made of videos from YouTube[1]. Because the type of collection inherently consists of various qualities, such as weather conditions and time of day, a two-step labeling including hand-extracted start and end time and performing a spatiotemporal annotation using VATIC [76] creates the annotations. CAD-CVIS [18] is another dataset collected from video-sharing websites. It uses LabelImg [77] to localize the accident in the frames of the videos.

3.2.2 Vehicle Trajectory Prediction Algorithms

Earlier models for trajectory prediction mostly used Recurrent Neural Networks such as Gated Recurrent Unit (GRU) or Long-short-term Memory (LSTM) for modeling the time dimension. CS-LSTM [16] has an LSTM-based encoder-decoder structure for embedding the vehicles' previous motions. On top of that, for modeling the interdependencies between vehicles, it uses a convolutional social pooling mechanism. GRIP++ [78] uses graphs to model the interactions between vehicles, and, using an LSTM encoder-decoder, it predicts the future trajectory. Later, attention

[1] www.youtube.com

Table 3.2: Comparison of existing camera view vehicle detection and tracking datasets with CHD. In the function column, D is for Detection, TP is for Trajectory Prediction, AD is for Accident Detection, and T is for Tracking. FPS stands for Frames Per Second. BE stands for Bird's-eye-view, DC stands for Dashcam view, HA and EL stand for High-angle and Eye-level view respectively.

Dataset	**Area**	**Len.**	**No of Frame**	**2D BB**	**Class**	**POV**	**Resolution**	**FPS**	**Aim**
TRANCOS	9	-	1244	46.8K	-	HA	360p	-	D
TRAF	20	39m	12K	19.9K	7	HA, DC	720p	10	TP
CADP	-	5.2h	518K	-	6	HA	Various	-	AD
CAD-CVIS	24	-	228K	-	-	HA	Various	-	AD
UA-DETRA	24	10h	140K	1.2M	2	HA	540p	25	D,T
CHD (Ours)	8	7.5h	1.6M	33.5M*	5	EY, HA	1080p	60	TP, D, T

** Bounding box annotations across all the recorded data in CHD.*

mechanisms found their way to trajectory prediction; using the Spatial-temporal Attention mechanism and LSTM units, STA-LSTM [79] improved the performance and explainability of trajectory prediction models. Trajectron++ [80] also has a recurrent structure based on LSTM and uses spatiotemporal graphs to model the input trajectories for vehicles and pedestrians. Like Trajectron++, Social-STGCNN [81] takes advantage of spatiotemporal graphs, but instead of the recurrent neural networks, it uses a convolutional structure and is primarily designed for pedestrian trajectory prediction.

DeepTrack[**?**] focused on real-world trajectory prediction applications such as traffic management and introduced a lightweight and agile model capable of real-time inference. Unlike previous methods, DeepTrack uses Temporal Convolutional Networks (TCNs) for modeling the time dimension. With Cyber-physical Systems (CPS) applications in mind, Pishgu [68] presents an efficient universal network architecture for trajectory prediction in different domains using Graph Isomorphism Networks (GINs) and convolutional attention mechanism. It is conventional for probabilistic trajectory prediction models to predict N number of possible future trajectories and pick the

best one [16, 81]. However, multiple predicted trajectories per subject in real-world scenarios are not very effective. Thus, more recent models such as [78, 79, 80, **?**, 68] primarily focus on single future trajectory prediction.

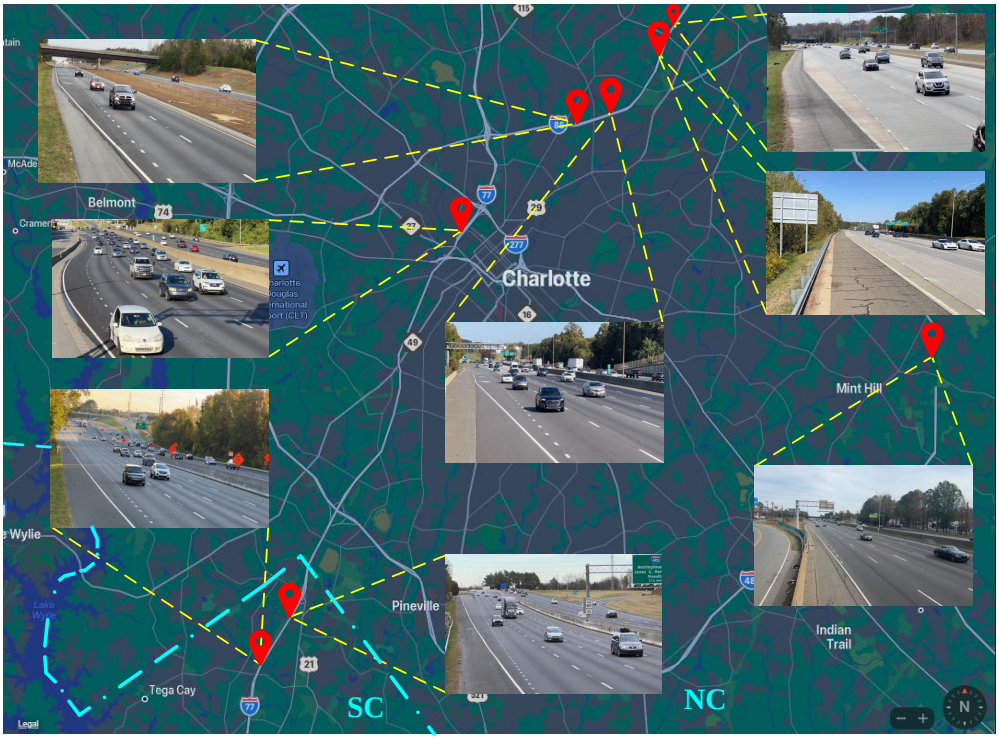

Figure 3.1: Spatial distribution of all the eight recording locations on the highways of NC and SC included in CHD.

3.3 Carolinas Highway Dataset (CHD)

CHD is a comprehensive collection of vehicle highway trajectories extracted from videos recorded across North Carolina and South Carolina from two distinct POVs: eye-level and high-angle. CHD also provides annotations for multi-vehicle detection and tracking. Fig. 3.1 shows the spatial map of all the recording locations with high-angle POV images of each site. The videos were recorded in full HD (1080p) resolution between 9 AM and 7 PM at eight locations within different localities.

The recording locations were selected to capture different traffic flows, vehicle behaviors, naturalistic driving patterns on highways with different road geometries, and varying lengths and structures of merging lanes. The videos were recorded at different

times of the day to capture various lighting conditions, traffic patterns, and volume. CHD captures data from multiple POVs, providing a diverse perspective of incoming traffic.

3.3.1 Annotations and Trajectory Extraction

One of the unique features of the proposed dataset, CHD, lies in the use of generated annotations, which reflect the annotations available in real-world applications where hand annotations are often unavailable [82]. Generated annotations may contain some noise or inaccuracies, but they provide a more realistic representation of annotations used in the real world. Using generated annotations, CHD can more accurately reflect the challenges in real-world applications, such as occlusions, varying lighting conditions, and other factors affecting detection and tracking accuracy.

CHD employs bounding box annotations to identify vehicles in each frame and spatially locate them. To generate high-quality bounding boxes, the CHD utilizes YOLOv5 [83]. Specifically, the YOLOv5x6 model has been trained on the BDD100k dataset [84] to identify different types of vehicles.

These bounding boxes provide crucial information that the tracking model uses to assign a unique identification number to each vehicle across multiple frames. CHD uses ByteTrack [85], in combination with YoloV5 tracks multiple vehicles in a frame. ByteTrack, with a bounding box as a basic tracking unit, uses data association to match the vehicles in different frames assigning them a unique ID.

In addition to utilizing the bounding box annotations to identify and locate vehicles in a given scene, the center of the bounding box is used to determine the position of each vehicle in the scene. This information is used to generate trajectories, which refer to the path followed by the object over a period of time. Specifically, in the context of vehicle tracking in CHD, the trajectories were generated by tracking the vehicle's position in successive video frames, using a unique ID assigned to each vehicle by ByteTrack as a unique trajectory in the dataset.

Extracted raw trajectory data was processed to improve the quality and accuracy of the data and to remove noise, outliers, or irrelevant information and to ensure that the analysis is based on the most relevant and reliable information. Unique trajectories of a minimum duration of four seconds and above were included in the dataset, enabling CHD for real-world models with smaller input and output windows. Additionally, stationary vehicles and vehicles moving away from the camera were filtered out, as the focus was on incoming traffic. False detections were also manually filtered out whenever possible to eliminate additional noise in the data.

3.3.2 CHD Statistics and Format

CHD consists of 338,000 trajectories extracted from 16 videos, with a total of 1.6 million frames. This is outlined in Table 3.1, demonstrating the scale of CHD comparable to that of well-known trajectory datasets. Regarding the number of bounding boxes for trajectory data, CHD aligns with widely used datasets, as shown in Table 3.1. Along with trajectory data, it includes raw videos and bounding box annotations for all the recorded videos. The 33M bounding box annotations reported in Table 3.2 showcase its high traffic density of recording environments.

In addition to the vast number of frames, CHD benefits from high-quality (1080p resolution) image data recorded at the frame rate of 60 fps, which exceeds others, as demonstrated in Table 3.2. As summarized in previous sections and Tables 3.1 and 3.2, CHD is among the few datasets with multiple POVs of incoming vehicles trajectory.

Ensuring consistency with commonly utilized trajectory prediction models [**?**, 68, 78], the trajectory data in CHD is extracted at a frame rate of 5 fps that was uniformly distributed with 70% assigned to the training set, 20% to the validation set, and 10% to the test set. The trajectory data are also extracted at 60 fps to facilitate research at higher frame rates. This dataset consists of five different classes of vehicles, and the distribution of different classes across different sets is presented in Fig. 3.2. It

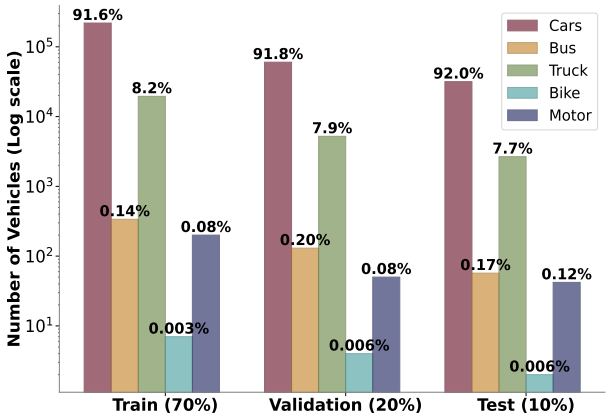

Figure 3.2: Vehicle distribution in the CHD trajectory data across test, train, and validation sets.

can be seen that CHD has around 90% cars, and the rest 9% is divided between bus, truck, bike, and motor category.UA-DETRAC[86] high-angle dataset from Tabel 3.2, exhibits similar distribution with around 87% of cars and remaining 13% distribution of other vehicles. Whereas, HighD[87] and NGSIM[1, 2] trajectory datasets from Tabel 3.1, has about 70% and 96% of cars.

CHD comprises several components for each recording, including raw videos, annotation data, and extracted trajectories. The annotation data file includes details such as the frame number, vehicle identification number, vehicle type, and bounding box coordinates. The vehicle trajectory data files contain frame and ID information and X and Y coordinates. Overall, the high frame rate, a large number of high-quality images and multiple POVs, and a comprehensive collection of real-world data make CHD a suitable benchmark for various ITS applications.

3.4 PishguVe

Vehicle trajectory prediction aims to forecast where an individual vehicle will be in the future, T time steps ahead, based on their past positions. The proposed model considers intrinsic and extrinsic factors influencing a vehicle's trajectory to achieve high accuracy while being efficient in real time. Using observed trajectories of all vehicles in a given scene, the model can accurately predict future positions by

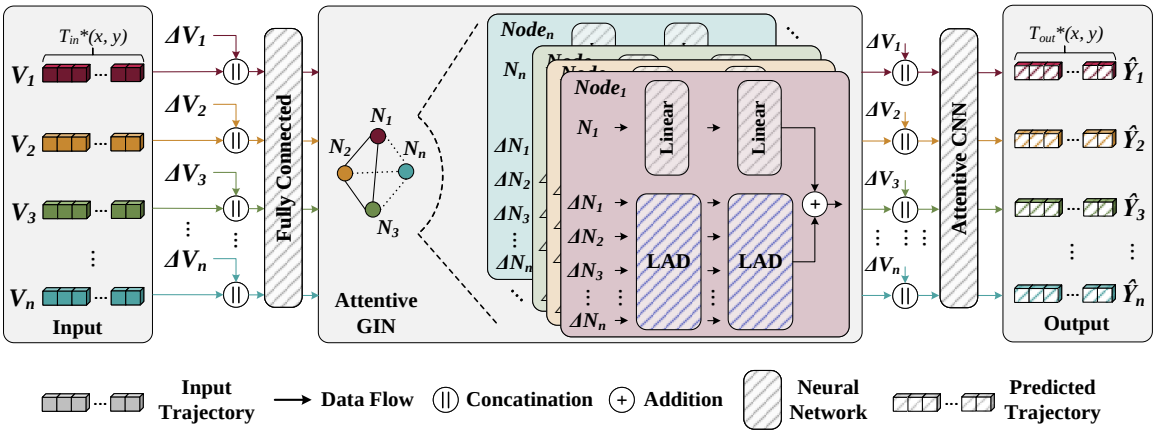

Figure 3.3: PishguVe architecture overview. Input trajectory vector, V_i (where i $\in$ (1 to n)) and relative trajectory vector ,ΔV_i (where i $\in$ (1 to n)) with x and y coordinates and input time window of T_{in} are concatenated and inferenced through the fully connected layer. The output of the fully connected layer, N_i, and neighbor aggregation vector ΔN_i is utilized by Attentive GIN in two separate branches for aggregating the node and neighbor features for detailed representation. The spatiotemporal attentive CNN is then used to predict the future trajectories of all vehicles up to T_{out} time steps ahead.

incorporating temporal and spatial dependencies between the vehicles.

In this study, we distinguish various elements of vehicle trajectory prediction as follows: The past trajectories of vehicles are represented by a set of absolute coordinates, denoted as V_i, and a set of relative coordinates, denoted as ΔV_i. The absolute coordinates are defined as $V_i = (x_i^t, y_i^t)$, where $t = 1, ..., T_{in}$, $i \in 1, 2, ..., n$ representing the index of the vehicle and x_i^t and y_i^t are x and y coordinates of the center of bounding box of vehicle i at time t. The relative coordinates are defined as $\Delta V_i = (x_i^t - x_i^1, y_i^t - y_i^1)$.

The study's objective is to predict the future trajectories of each vehicle, $\hat{Y}_i$, using its past trajectories as input. The predicted future trajectories are represented as $\hat{Y}_i = (x_i^t, y_i^t)$, where $t = (T_{in} + 1), ..., T_{out}$ and $i \in 1, 2, ..., n$, are generated as a set of coordinates for each vehicle, indicating their positions for future time steps. These predictions are evaluated by comparing them with the ground truth future trajectories, denoted as Y.

3.4.1 Network Architecture

The overall architecture of Pishgu-Ve can be seen in Fig. 3.3. In the first step, the absolute and the relative coordinates are concatenated together and passed through a fully connected layer to transform into the latent space:

$$N_i = f\left(W_1.(V_i \parallel \Delta V_i) + B_1\right) \tag{3.1}$$

Where N_i, f, W_1, and B_1 stand for the embedded representation of the i^{th} vehicle in the scene, the Leaky ReLU activation function, and the corresponding weights and the biases of the fully connected layer accordingly. V_i and ΔV_i are absolute and relative trajectories, and $||$ represents concatenation. The absolute coordinates show the global position of each vehicle. In contrast, the relative coordinates provide information about how each vehicle moves with respect to its past trajectory, and the single fully connected layer integrates both into a latent representation.

N_i is then fed to the Graph Isomorphism Network (GIN) for capturing the inter-dependencies between available vehicles in the scene. Each vehicle is represented as a node in a fully connected graph. The graph is chosen to be fully connected to remove predefined biases and allow the network to decide how much data should be incorporated into the output in the message-passing process. On the other hand, having a fully connected graph enables PishguVe to combine features across all nodes in a single graph operation which helps with the efficiency of the architecture. Similar to [68], we adopt two separate networks for aggregating the node and neighbor features to create a richer presentation. For the node features, we utilize a fully connected network with one hidden layer:

$$G_i' = W_3.(W_2.(1+\theta)N_i + B_2) + B_3 \tag{3.2}$$

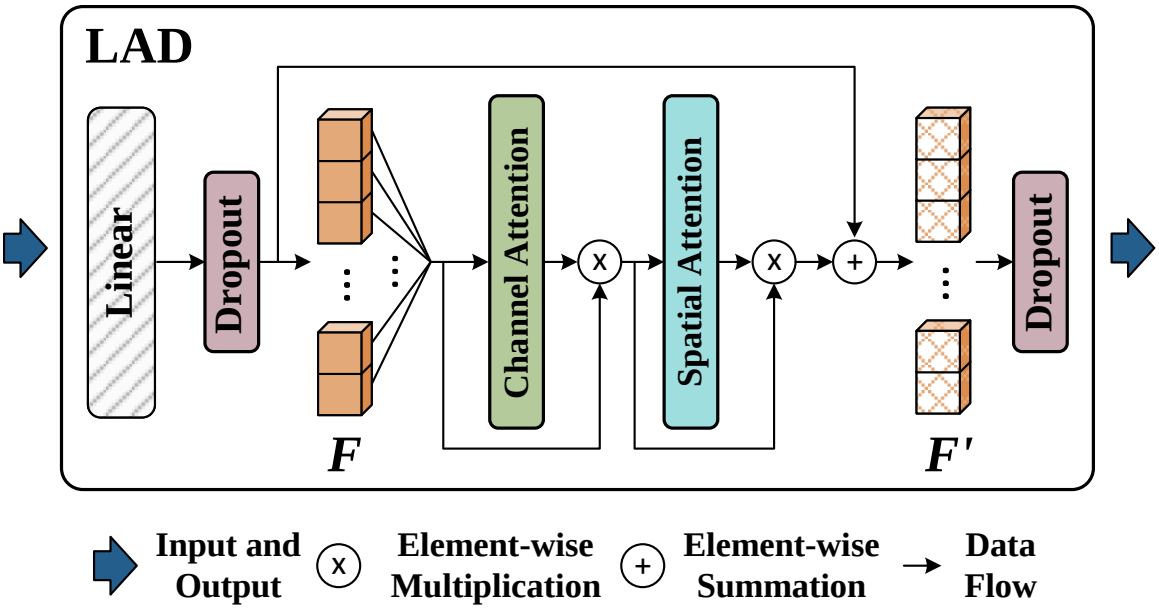

Figure 3.4: The Linear Attention Dropout (LAD) Block. LAD is integrated into GIN for feature aggregation using channel and spatial attention. Dropout layers are added after linear and attention layers to minimize overfitting.

Where G'_i is the i^{th} node aggregated features, W_2, B_2, W_3, and B_3 are the parameters of the two consecutive fully connected layers and θ is a learnable parameter. In contrast to [68], for neighbor aggregation, we define a new block called Linear Attention Dropout.

Linear Attention Dropout or LAD block is designed explicitly for neighbor feature aggregation. Fig. 3.4 shows the details of the LAD block. LAD consists of a linear layer followed by attention layers for highlighting the most informative neighbor features to construct a richer representation. We adopt channel, and spatial attention from [88] to integrate attention efficiently into the GIN. Also, to avoid overfitting, we leverage two dropout layers. Two back-to-back blocks of LAD are used for neighbor aggregation for extracting higher-level features. The LAD operation can be summarized as follows:

$$\begin{aligned} F'_j = LAD(N_j) = & MLP_1(Pool_{avg}(W_4.\Delta N_j + B_4) + \\ & MLP_1(Pool_{max}(W_4.\Delta N_j + B_4)) \end{aligned} \tag{3.3}$$

Where W_4 and B_4 are the parameters of the linear layer, MLP_1 is a shared Multi-layer Perceptron with one hidden layer, and F'_j is the output of the LAD block. The final output of GIN will be constructed by the sum of outputs of the node aggregation and neighbor aggregation networks, which can be described as:

$$GIN(N_i) = G'_i + \sum_{j \in \mathcal{V}(i)} LAD(F'_j) \tag{3.4}$$

Where $\mathcal{V}(i)$ represents the set of neighbors for the i^{th} node.

In the next step, the enriched feature maps are concatenated with relative coordinates again to emphasize the relative movement compared to previous time steps and fed to an attentive CNN for the final trajectory prediction. CNNs have shown great capacity for extracting powerful feature maps. On top of that, we again use an efficient channel and spatial attention mechanism to improve the capability of finding more influential features. Keeping efficiency in mind, PishguVe has three convolutional layers followed by attention blocks and a last 1×1 convolutional layer for forming the final output. The first convolutional layer has a kernel size of 2×2 for capturing the low-frequency patterns, and the following two convolutional layers have a kernel size of 2×1.

3.5 Evaluation and Experiments

PishguVe is evaluated on the widely used NGSIM dataset [2, 1] and CHD dataset proposed in this manuscript. Similar to previous works[16, 78, 89], the 8 million data entries of the NGSIM were distributed 70% training, 10% validation, and 20% testing sets. All models evaluated on CHD use a the data split discussed in Section 3.3.2. All experiments were carried out on a Workstation equipped with a Threadripper Pro 3975WX processor with 32 cores clocking at 3.50 GHz and three A6000 GPUs.

3.5.1 Evaluation Metrics

Root Mean Square Error (RMSE), Average Displacement Error (ADE), Final Displacement Error (FDE), and the number of model parameters are used as a measure of prediction accuracy and performance of the models. The error definitions are illustrated in Fig. 3.5, visually representing the concepts under consideration. The figure assists in understanding each definition's nuances and highlighting their similarities and differences.

Root Mean Square Error (RMSE) is commonly used to evaluate the accuracy of a predictive model that estimates the trajectories of vehicles in a scene. At a given time point, t, the RMSE is calculated as the square root of the mean square error between the predicted path ($\hat{Y}$) and the ground truth path (Y) of the n subjects of interest in the scene:

$$\text{RMSE}^t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i^t - \hat{Y}_i^t)^2} \tag{3.5}$$

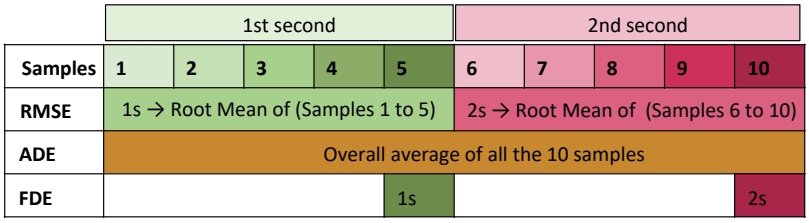

Figure 3.5: Error matrices visualization for two seconds time window and a data rate of five samples/second.

Average Displacement Error (ADE), in the context of predicting the trajectories of vehicles in a scene, is an evaluation metric that measures the distance between the predicted ($\hat{Y}$) and the ground truth coordinates (Y) over all T_{out} predicted time steps and all subjects of interest (n) in the scene:

Table 3.3: Performance comparison of vehicle trajectory prediction approaches on NGSIM datasets[1, 2].

Model	**ADE (m)**	**FDE (m)**	**1s**	**2s**	**3s**	**4s**	**5s**	**Params (K)**
CS-LSTM[16]	2.29	3.34	0.61	1.27	2.09	3.1	4.37	191
GRIP++[78]	2.01	3.25	0.38	0.89	1.45	2.14	2.94	-
STA-LSTM[79]	1.89	3.16	0.37	0.98	1.71	2.63	3.78	124
DeepTrack[**?**]	2.01	3.21	0.47	1.08	1.83	2.75	3.89	**109**
Pishgu[68]	0.88	1.96	0.15	0.46	0.82	1.25	1.74	132
PishguVe (Ours)	**0.77**	**1.76**	**0.11**	**0.37**	**0.70**	**1.09**	**1.55**	133.5

$$\text{ADE} = \frac{1}{n * T_{out}} \sum_{i=1}^{n} \sum_{t=1}^{T_{out}} \left| Y_i^t - \hat{Y}_i^t \right|_2 \tag{3.6}$$

The $|\cdot|$ 2 notation denotes the L2 norm, which calculates the Euclidean distance between the predicted and actual coordinates. The ADE measures the average displacement error between the predicted and actual coordinates of the subjects of interest in the scene. A lower ADE indicates better accuracy of the predictive model.

Final Displacement Error (FDE) is an evaluation metric that measures the L2 distance between the predicted coordinates ($\hat{Y}$) and the ground truth coordinates (Y) of the last predicted time step for all subjects of interest (N) in the scene:

$$\text{FDE} = \frac{1}{n} \sum_{i=1}^{n} \left| Y_i^{T_{out}} - \hat{Y} i^{T_{out}} \right|_2 \tag{3.7}$$

Parameters refer to the weights and biases of a neural network learned during training. The number of parameters affects models' performance and generalization ability.

Table 3.4: Average and final displacement error-based performance of PishguVe with varying dropout probabilities.

Attention Dropout	**Linear Dropout**	**ADE**	**FDE**
0.03	0.020	17.84	63.14
0.10	0.020	17.53	62.14
0.20	0.020	17.74	63.01
0.25	0.020	**16.81**	**57.71**
0.40	0.020	17.36	61.90
0.25	0.025	17.44	61.46
0.25	0.15	17.86	62.22
0.40	0.15	17.82	62.59
0.40	0.30	18.78	68.51

3.5.2 Hyperparameter Search

In this work, we extend the Graph Isomorphism Network (GIN) by adding attention mechanisms to improve performance. To evaluate the effectiveness of our approach, we conducted ablation studies and experiments such as adding attention at the node level in the Attentive GIN block, adding the LAD block at the node level, the relative nodes level, and both levels together. As expected, the best performance improvement was achieved by adding attention to the relative node level, which was missing a learnable parameter. However, overfitting was observed when the model was trained over a few epochs, as testing errors (i.e., ADE and FDE) kept rising with decreasing training errors. To overcome the overfitting, dropout layers were applied, as shown in Fig. 3.4. Table 3.4 shows the training ADE and FDE for PishguVe with different dropout probabilities. The tests were performed using the CHD High-Angle data and then extended to other datasets. Table 3.4 only shows the tests performed on the CHD High-Angle dataset, as it best represents the behavior of varying dropout values on PishguVe.

The optimal dropout probabilities of the attention and linear layers were determined through experimentation, with the best ADE and FDE performance for probabilities of 0.25 and 0.02, respectively. The placement of dropout layers is shown

Table 3.5: Performance comparison of vehicle trajectory prediction approaches on CHD eye-level POV data.

Model	**ADE (pixels)**	**FDE (pixels)**	**1s**	**2s**	**3s**	**4s**	**5s**	**Params (K)**
Social-STGCNN[81]	24.33	95.22	4.32	9.15	15.93	29.05	68.32	**7.4**
GRIP++ [78]	44.27	129.58	4.42	12.86	24.31	35.04	145.17	-
Pishgu [68]	37.99	123.69	4.98	13.58	26.61	50.31	106.45	132
PishguVe (Ours)	**20.75**	**69.33**	**3.21**	**8.24**	**15.55**	**28.46**	**57.88**	133.5

in Fig. 3.4, and the performance of PishguVe with varying dropout probabilities is outlined in Table 3.4.

All subsequent evaluations and comparisons of PishguVe presented in this manuscript were performed using the earlier probabilities for attention and linear dropout. These probabilities were selected to strike a balance between reducing overfitting and preserving the accuracy of the proposed model.

3.5.3 Results

This section evaluates the proposed PishguVe model against existing approaches on multiple datasets. Firstly, PishguVe is evaluated on the NGSIM dataset, including analysis and comparisons with existing approaches, as shown in Table 3.3.

Next, performance PishguVe is also accessed on the proposed Carolinas Highway Dataset, with eye-level and high-angle POV data. This assessment excludes explicit models using lane number as one of the features as it is representative of real-world applications while still achieving best-in-class results.

3.5.3.1 NGSIM-based trajectory prediction

This section assesses the performance of PishguVe, by comparing its accuracy with several current vehicle trajectory prediction models on the NGSIM datasets [1, 2]. The first model we compare with is the Convolutional-Social LSTM (CS-LSTM)[16], an encoder-decoder model that captures vehicle interactions to predict multiple trajecto-

ries for the ego vehicle. GRIP++[78] is a graph-based approach that leverages vehicle-environment interactions to generate more accurate trajectory predictions. Another model we consider is the Spatiotemporal Attention-LSTM (STA-LSTM)[79], which incorporates both spatial and temporal information and uses an attention mechanism to weigh the influence of historical trajectories and neighboring vehicles on the ego vehicle. The DeepTrack[**?**] model is a temporal convolution networks-based encoder-decoder architecture that uses attention to predict a single trajectory for the ego vehicle. Finally, Pishgu[68] is a GIN-based vehicle trajectory prediction model that employs an attentive CNN for prediction. The following sections present a thorough comparative analysis of PishguVe's performance against these models.

Table 3.3 shows that PishguVe performs better than all the selected models and achieves the lowest RMSE (at 1,2,3,4 and 5th seconds), ADE, and FDE values. However, DeepTrack has the least parameters, with STA-LSTM being the close second. PishguVe performs 12.50% and 10.20% better in ADE and FDE than the current SotA, Pishgu. It also performs better in terms of RMSE at each time step. Finally, we note that PishguVe has slightly more parameters (133.5K) than Pishgu (132K) but fewer parameters than CS-LSTM and GRIP++. This indicates that PishguVe achieves superior performance with a reasonable number of model parameters, making it an attractive choice for practical applications. PishguVe also achieves a minor but significant improvement over the Pishgu model. These results highlight the effectiveness of the proposed PishguVe architecture for vehicle trajectory prediction on the NGSIM dataset.

3.5.3.2 CHD for trajectory prediction pov

For trajectory prediction on the CHD datasets, we evaluated the performance of PishguVe against three contemporary models that do not utilize lane number as an input feature, GRIP++ [78], Pishgu [68], and Social-STGCNN [81]. The exclusion of lane number as a feature is motivated by challenges faced by real-world systems,

Table 3.6: Performance comparison of vehicle trajectory prediction approaches on CHD high-angle POV data.

Model	**ADE (pixels)**	**FDE (pixels)**	**1s**	**2s**	**3s**	**4s**	**5s**	**Params (K)**
Social-STGCNN [81]	31.87	98.46	9.74	21.83	29.01	42.34	82.14	**7.4**
GRIP++ [78]	36.32	100.89	**3.40**	**6.67**	14.32	28.02	123.04	-
Pishgu [68]	<u>18.33</u>	<u>61.92</u>	4.04	7.48	<u>13.99</u>	<u>24.30</u>	<u>51.51</u>	<u>132</u>
PishguVe (Ours)	**16.81**	**57.71**	<u>3.52</u>	<u>7.12</u>	**12.64**	**22.93**	**48.60**	133.5

such as inaccuracies and errors in lane detection propagating to system output and sensitivity of lane detection approaches to external factors such as weather.

GRIP++ and Pishgu were also used for comparisons on the NGSIM dataset. The Social-STGCNN is included here due to its extremely low complexity and best-in-class performance in pedestrian trajectory prediction. It is used for modeling the spatiotemporal patterns of human movements in social groups. Each model was trained multiple times for several epochs on eye-level and high-angle datasets to obtain the best results. By comparing PishguVe's performance against these models, we aim to demonstrate the effectiveness of the proposed model in improving trajectory prediction accuracy while reducing the reliance on lane identification.

CHD Eye-Level POV. Table 3.5 outlines the performance of the aforementioned models on the CHD Eye-Level dataset. PishguVe outperforms all other models with an ADE of 20.75 pixels and an FDE of 69.33 pixels, respectively. Specifically, compared to the second-best model, Social-STGCNN, PishguVe achieves a 14.58% lower ADE and a 27.38% lower FDE.

Regarding prediction accuracy at different time horizons, PishguVe outperforms all other models, achieving the lowest RMSE values. Compared to Social-STGCNN achieves 25.74%, 10.09%, 2.40%, 18.48%, and 15.85% lower RMSE values at 1s, 2s, 3s, 4s, and 5s, respectively. These results suggest that PishguVe is more accurate and effective for predicting vehicle paths than other SotA models for eye-level and birds-

eye-view datasets. However, PishguVe reports the highest number of parameters, 133.5K, marginally behind the second-best model Pishgu with 132K parameters.

CHD High-Angle POV. The proposed PishguVe model was also trained on the CHD High-Angle dataset, and the performance results presented in Table 3.6 were obtained using this dataset. PishguVe achieves an ADE of 16.81 pixels and an FDE of 57.71 pixels, which are lower than the other models. The second best model for ADE and FDE is Pishgu, with an ADE of 18.33 pixels and an FDE of 61.92 pixels. PishguVe achieves an 8.3% improvement in ADE and a 6.9% improvement in FDE over the second-best model.

PishguVe also performs well in terms of RMSE for certain time thresholds. It achieves the lowest error rates for 3s, 4s, and 5s and is slightly lower than GRIP++ for 1s and 2s. Compared to the second-best model, Pishgu, PishguVe achieves a 13.9%, 5.1%, 0.7%, 9.6%, and 5.8% improvement for 1s, 2s, 3s, 4s, and 5s, respectively.

Furthermore, PishguVe outperforms the Social-STGCNN model, which has relatively fewer parameters but struggles in generalizing results. The ADE and FDE performance of PishguVe in the High-Angle POV dataset is 89.36% and 70.94%, respectively. Moreover, for root mean square error (RMSE) comparisons to Social-STGCNN, PishguVe performs better by 178.29%, 206.52%, 129.10%, 84.74%, and 68.99% for 1s, 2s, 3s, 4s, and 5s, respectively.

3.6 Conclusion

This paper presents the *Carolinas Highway dataset (CHD)*, which consists of over 338,000 vehicle trajectories captured from 1.6M high-resolution images recorded at eight highway locations from two distinct POVs. The proposed dataset provides a unique benchmark for evaluating trajectory prediction and various highway-based applications in ITS.

We also introduce *PishguVe*, a SotA trajectory prediction architecture that utilizes graph-based attention and an attentive neural network to extract essential features

and produce real-time results. It creates graphs of all vehicles in a scene to learn their dependency and behavior in different driving conditions. Our experiments on the NGSIM and CHD have demonstrated that PishguVe sets a new state-of-the-art for vehicle trajectory prediction in bird's-eye, high-angle, and eye-level POV, achieving better performance compared to existing approaches. The proposed architecture and dataset can facilitate the development of more advanced driver safety and assistance systems, intelligent transportation systems, and traffic analysis and surveillance tools, significantly improving road safety and efficiency.

CHAPTER 4: VEGAEDGE: EDGE AI CONFLUENCE ANOMALY DETECTION FOR REAL-TIME HIGHWAY IOT-APPLICATIONS

4.1 Introduction

In today's digital age dominated by the Internet of Things (IoT), camera-based infrastructure has become an integral part of our interconnected world. With urbanization intensifying, our highways face increasing congestion and unpredictable driving patterns. Although current highway cameras offer surveillance, their true potential to harness real-time analytics remains largely untapped. Integrating edge-based AI frameworks with these cameras can revolutionize traffic management and safety [63]. This integration not only promises rapid detection and response to anomalies but also increases bandwidth efficiency, lowers latency, and scales highway monitoring, marking a transformative approach to road safety and management.

The AI-based edge applications can help with real-time detection of erratic driving behaviors that can help tackle the distressing surge in accidents, especially within work zones. From 2003 to 2020, worker fatalities rose, with 135 deaths in 2019 and 117 in 2020 [22, 23]. The Federal Highway Administration's 2021 report highlighted 106,000 work zone accidents, resulting in 42,000 injuries and 956 fatalities [5, 90]. While traditional safety mechanisms in these zones are primarily reactive [91], often leading to late interventions, integrating AI at the data's edge ensures timely decision-making crucial for highway safety, surveillance, and traffic analysis applications.

Anomaly detection for roadways mainly focuses on the complexities of autonomous driving [24] in urban settings, where interactions among vehicles, infrastructure, and pedestrians are intricate. Influenced by factors like intersections and diverse road alignments, urban trajectories are notably unpredictable. In contrast, highway travel,

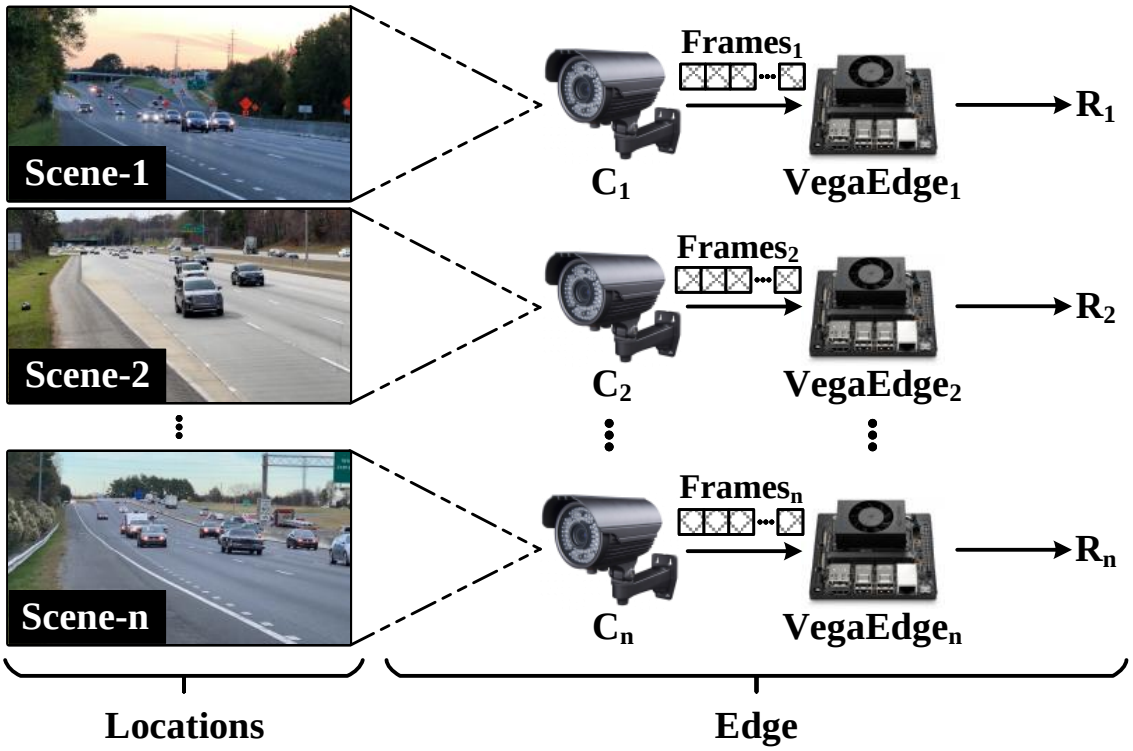

Figure 4.1: Example demonstration of VegaEdge implementation on an IoT platform for real-world highway scenarios.

designed for longer distances, exhibits more predictable behaviors [92, 93, 94, 95].

For effective AI in highway settings, models need training on specific datasets differentiating normal from abnormal driving. Many existing datasets, however, lack resolution and relevance. Addressing this, we present the *Carolinas Anomaly Dataset (CAD)* with real-world highway anomalies. Moreover, real-time anomaly detection is crucial for edge-based safety applications prioritizing nimbleness. CAD emphasizes identifying vehicle trajectories that deviate from standard paths, especially those moving outside of their lanes. While many anomaly detection methods exist [96, 97, 98], they're not highway-specific and require significant computational power. Overall, there is a clear deficiency in real-world highway anomaly datasets and corresponding algorithms, leading to a void in AI frameworks for real-time anomaly detection in practical applications.

In this context, we introduce *VegaEdge*, an edge AI confluence tailored for real-

time highway IoT applications operating on embedded edge devices using lightweight anomaly detection. Fig. 4.1 shows how VegaEdge can monitor highway traffic and detect anomalies at the edge at various locations on embedded platforms. Our evaluations on real-world datasets confirm VegaEdge's effectiveness and its performance with real-world video detection will be detailed in subsequent sections. In this paper, we also introduce a lightweight method that uses ground truth and trajectory prediction for quick anomaly detection, promoting enhanced highway safety. This method seamlessly integrates with the State-of-the-Art (SotA) trajectory prediction [25] integrated with VegaEdge providing swift real-world anomaly detection.

We extensively evaluated VegaEdge across three distinct platforms, two of which are edge-based, low-power devices, underscoring its versatility. Tests were conducted to validate its robustness using real-world and simulated videos, demonstrating VegaEdge's capability to function with digital twins and across varying traffic densities. Furthermore, we examined its performance specifically for highway work zone safety, analyzing the impact of diverse prediction windows on the buffer times afforded to workers during potential hazards. The efficacy of our anomaly detection is showcased through evaluations on both adversarial and real-world datasets. These tests underscore the pronounced differences between adversarial-generated trajectories and real-world scenarios, emphasizing the imperative of employing real-world videos in authentic system deployments. We also perform extensive power analysis on an embedded to provide insights into the power consumption of VegaEdge in different power modes that can be utilized based on the desired application.

The main contributions of this paper are summarized as follows:

- We introduce Carolinas Anomaly Dataset (CAD), a new real-world anomaly dataset for highway applications. This dataset empowers researchers to validate anomaly detection techniques within genuine highway contexts.

- We present a novel anomaly detection technique that foresees anomalous driving

behaviors out of the predicted trajectories by extrapolating angle-based and displacement errors. Its effectiveness is demonstrated with adversarial and real-world trajectories on select datasets.

- We introduce VegaEdge, a cutting-edge AI-powered IoT solution for vehicle anomaly detection designed for edge-based embedded systems. It is adept at identifying vehicles that diverge from their anticipated route, indicating possible hazardous intrusions on highways in real-time.

- We subject VegaEdge and proposed anomaly detection techniques to exhaustive evaluations across multiple platforms and scenarios. The results showcase its adaptability and superior performance in real-world and simulated environments. We also demonstrate its effectiveness, emphasizing its application in work zone safety.

4.2 Related Works

Efforts have been made to adapt anomaly and vision models for IoT devices. [99] presents an IoT-focused video surveillance system, primarily analyzing human-related events. [100] explores vision model applications in IoT, while [101] investigates anomaly detection in time series data for domains like smart cities. Recently, there's been increased focus on highway safety. [102] introduces a trajectory prediction framework for dense traffic, utilizing LSTMs and CNNs. [103] uses road geometry for vehicle counting, speed estimation, and classification. [104] suggests a real-time flow estimation system based on pairwise scoring for vehicle counting. MultEYE [105] is an aerial viewpoint vehicle tracking system, leveraging segmentation for detection accuracy in edge devices and IoT applications. Nevertheless, a notable gap persists within AI-based solutions for highway applications, primarily due to the limited availability of real-world datasets and dedicated frameworks tailored specifically to highway-based edge applications with real-time processing capabilities.

Anomaly detection in vehicle frameworks has been explored in various studies. [96] proposes an IoT system detecting abnormal driving using semantic analysis, vehicle detection, and 5G communication. [106] employs re-identification and multi-camera tracking with Gaussian Mixture Models (GMMs) to analyze vehicles. Anomalies are identified based on foreground-background changes. [97] offers a tracking algorithm for anomaly detection in road scenes. [107] enhances vehicle anomaly detection accuracy by integrating road geometry with movement predictions. [98] presents a multi-granularity design combining various tracking levels for vehicle anomalies. Using clustering, [108] introduces a probabilistic framework for anomaly detection via vehicle trajectories.

Several studies, such as DSAB [24], focus on the vehicle anomaly detection problem individually. DSAB reconstructs vehicle social graphs using the Recurrent Graph Attention Network. [109] employs Graph Convolutional Networks (GCNs) with a contrastive encoder for feature extraction, with the features later used in an SVM classifier. They also explore unsupervised methods using an Adversarial Autoencoder.

While there's a scarcity of comprehensive vehicle datasets in highway safety due to data gathering challenges, AI City Challenge offers a benchmark [110, 111]. Still, its alignment with highway safety is limited. The Carolinas Highway Dataset (CHD) [25] provides videos from multiple viewpoints, ideal for highway safety. Given the rarity of anomalous driving behaviors, [112] suggests an adversarial framework to generate anomalies on existing datasets. Recent studies like [109] are utilizing this approach for more exhaustive anomaly detection evaluations. This lack of resources and use of adversarial approaches underscores the urgency of developing and advancing real-world datasets and AI-based IoT-edge solutions that capable of handling the unique challenges and anomalies of highway safety and surveillance.

4.3 CAD: Carolinas Anomaly Dataset

Building upon discussions from prior sections, the absence of dedicated highway-based trajectory anomaly datasets presents a challenge in validating our anomaly detection methodologies. To circumvent this limitation, we adopt the recent advancements in adversarial anomaly generation as a testing bed for our proposed approach. Zhang et al. [112], introduce an adversarial attack-based technique designed to craft realistic anomalous trajectories by perturbing standard trajectories within a dataset.

While adversarial approaches have made significant strides in improving the fidelity of generated results, they still fall short of perfectly mirroring real-world scenarios. Although the disparity between machine-generated anomalies and actual real-world anomalies has diminished, it has not been completely eradicated. In light of the challenges inherent in evaluating highway anomalies and the existing gap in relevant datasets, we present the "Carolinas Anomaly Dataset (CAD)". This dataset, derived from the Carolinas Highway Dataset (CHD) [25], encompasses 22 videos, each exhibiting at least one anomalous driving trajectory. These videos are captured from two distinct vantage points: high-angle and eye-level, offering a versatile tool for surveillance and road safety applications. Specifically, CAD is composed of one-minute video segments, evenly split between the two perspectives, showcasing variety of anomaly behavior.

In this context, anomalous behavior pertain to the atypical movement patterns exhibited by vehicles, including actions such as vehicles deviating from their designated lanes on the highway, abruptly halting in front of the camera's view, or vehicles that approach the camera while diverging away from their designated lane. These unusual and non-standard behaviors have the potential to pose significant risks to nearby structures, infrastructure, and, most critically, to the safety of workers, particularly within the dynamic and often high-speed environment of highway work zones.Designed to enable the evaluation of various anomaly detection methodologies,

CAD serves as an invaluable resource for researchers focused on innovating highway safety through anomaly detection algorithms.

4.4 Anomaly detection Methodology

In this section, we present our methods for anomaly detection using predicted trajectories. For anomaly detection, the trajectory prediction output is used to evaluate the error and angle-based approaches. The goal was to evaluate both methodologies for detecting anomalous behavior in trajectory and video datasets. Through this methodology, we allow the detection of unusual vehicle behaviors, such as sudden lane changes, erratic driving, or potential security threats in desired applications with minimum computation.

4.4.1 ADE-based Anomaly Detection

Average Displacement Error (ADE) based Anomaly Detection is a method of computing the average error from the predicted trajectory to assess the accuracy of trajectory predictions for vehicles and comparing it against a threshold ($T^{ADE}_{anomaly}$) as per the application. It computes the average Euclidean distance between predicted trajectories ($\hat{F}$) and actual trajectories (P) overall predicted time steps (T_{pred}) and subjects (n) in a scene. The ADE in predicted trajectory from last T_{past} seconds is compared with the desired ADE threshold, $T^{ADE}_{anomaly}$ as:

$$\frac{1}{n * T_{past}} \sum_{v=1}^{n} \sum_{t=1}^{T_{past}} \left| \hat{F}_v^t - P_v^t \right|_2 > T^{ADE}_{anomaly}, \tag{4.1}$$

where $t = 1, ..., T_{in}$ is time and, $v \in 1, 2, ..., n$ representing the index of the vehicle. By setting a threshold, a criterion is set to identify anomalous trajectories. Value exceeding the threshold indicates a significant disparity between the predicted and ground truth trajectories, resulting from unexpected driving behavior, going off the lane, etc. As PishguVe is designed to predict the normal trajectory of the ego vehicle,

the ADE for any vehicle that exceeds the threshold is marked as an anomaly.

4.4.2 Angle-based Anomaly Detection

Angle-based Anomaly Detection calculates the angle between the predicted future trajectory vector, $\hat{F}_v$ and the actual trajectory vector, P_v of the ego vehicle and compares it with a threshold according to the application. Given the x and y coordinates of $\hat{F}_v$ and P_v for past few seconds t_{past}, we can compute the direction vectors:

$$D_{P_v} = \begin{bmatrix} x_{v,p}^{t_{past}} - x_{v,p}^{0} \\ y_{v,p}^{t_{past}} - y_{v,p}^{0} \end{bmatrix} \quad D_{\hat{F}_v} = \begin{bmatrix} x_{v,f}^{t_{past}} - x_{v,f}^{0} \\ y_{v,f}^{t_{past}} - y_{v,f}^{0} \end{bmatrix} \tag{4.2}$$

, here $x_{v,f}^{0}$ and $x_{v,f}^{0}$ are the position of vehicle a frame before the start of prediction. The angle between these direction vectors D_{P_v} and $D_{\hat{F}_v}$ is compared with the threshold, $T_{anomaly}^{Angle}$ as:

$$\arccos\left(\frac{D_{P_v} \cdot D_{\hat{F}_v}}{\|D_{P_v}\|\|D_{\hat{F}_v}\|}\right) > T_{anomaly}^{Angle}, \tag{4.3}$$

where $D_{P_v} \cdot D_{\hat{F}_v}$ is the dot product of D_{P_v} and $D_{\hat{F}_v}$, and $\|D_{P_v}\|$ and $\|D_{\hat{F}_v}\|$ are their respective magnitudes.

4.5 VegaEdge Design

VegaEdge is an integration of high-performing AI models to empower IoT-embedded edge devices. Specifically designed to enhance real-time safety and surveillance on highways, its core capabilities encompass vehicle detection, tracking, and trajectory prediction, all converging toward the final goal of anomaly detection. Fig. 4.2 provides a step-by-step visual representation, illustrating how the entire VegaEdge system operates in a union. The high-level pseudocode of VegaEdge is also shown in Algorithm

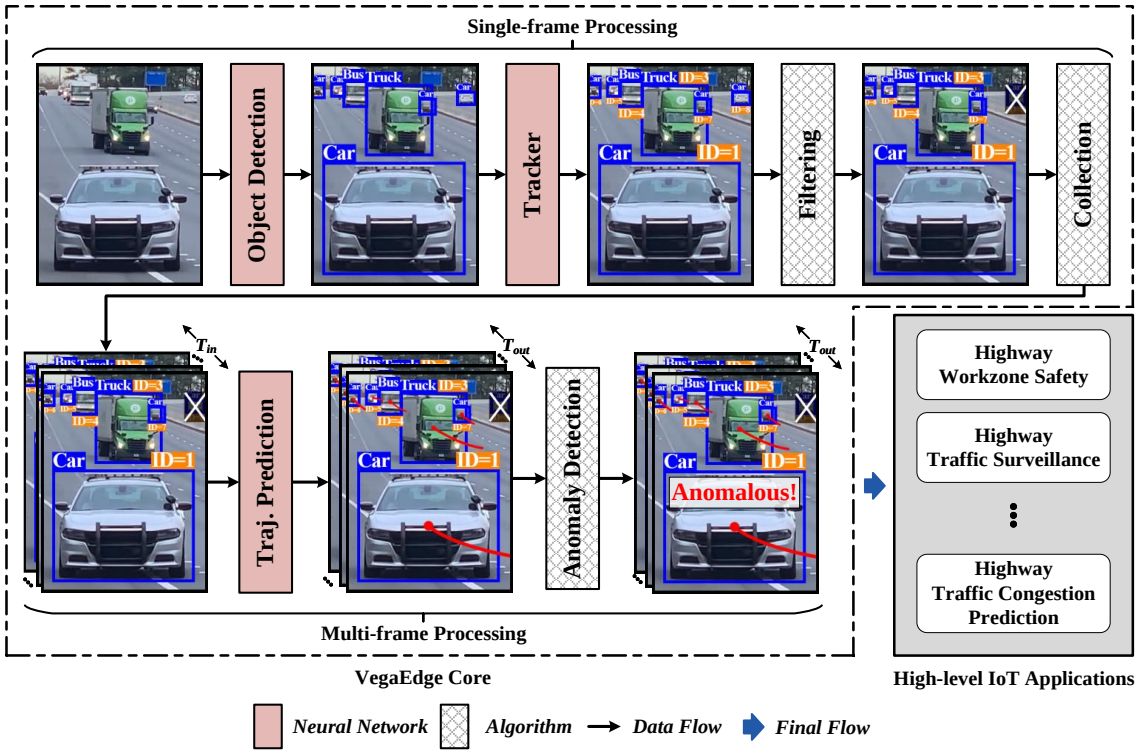

Figure 4.2: Data flow and algorithmic design of VegaEdge. Object detector and tracking process the frames in a single-frame manner. The filtering algorithm ensures that the focus is on the correct side of the road, and the Collection algorithm collects the sufficient number of frames T_{in} required by the Trajectory Prediction algorithm. The predicted path for T_{out} frames in comparison with the actual path will be used for Anomaly Detection, which finally can be used for highway IoT applications at the edge.

2.

As shown in Fig. 4.2 at a high level, VegeEdge detects vehicles within an image and subsequently tracks them across consecutive frames on a frame-by-frame basis. Following this, the system filters and accumulates the trajectories of distinct vehicles identified in the prior phase. Lastly, leveraging the gathered vehicle data from the past 3 seconds, it projects the trajectories for the upcoming five seconds, utilizing this foresight for effective anomaly detection. In the following subsections, we discuss our design choices and the working of each step shown in Fig. 4.2.

Algorithm 2 High-Level Workflow of VegaEdge

Require: RGB image
Ensure: Detected anomalous vehicle trajectories

Initialize: Detection, tracking, and prediction AI models
Warm-up for 3 seconds for vehicle detection and ReID
loop
    if frame is available **then**
        Detect vehicles in frames
        Assign unique IDs
        Ensure consistent IDs for previously identified vehicles
    else
        Read the next frame
    end if
    Set `inference_flag` to 0
    for each unique vehicle ID in frame **do**
        Remove vehicles moving away from the camera
        Remove IDs with less than 3 sec (15 frames) trajectory
        if vehicles available for inference (3 sec trajectory) **then**
            Set `inference_flag` to 1
        end if
    end for
    if `inference_flag` == 1 **then**
        `PishguVe` (Predict trajectory for next 5 seconds)
        if trajectory available for anomaly **then**
            for each Vehicle ID available **do**
                if Anomaly Criteria $> T_{\text{anomaly}}$ **then**
                    Flag anomaly detected for specific vehicle ID
                end if
            end for
        end if
    end if
end loop

4.5.1 Vehicle Detection

For efficient and rapid vehicle detection for edge-integrated IoT devices, we opt for the YOLOv8l [113] model and trained it on the BDD100k vehicle dataset [114]. Our decision was influenced by the system's overall performance, such as latency, accuracy, and memory requirements.

Model size and performance are critical in edge deployments, particularly for IoT devices. YOLOv8l addresses this by being 35.9% smaller than YOLOv8x [113], making it ideal for resource-limited embedded-IoT devices. Despite its reduced size, its mean Average Precision (mAP) is a competitive 52.9%, only 1% less than the 53.9% of YOLOv8x.

4.5.2 Vehicle Tracking

Multi-object tracking (MOT) is fundamentally concerned with the identities of objects within video sequences. ByteTrack [85] uses an innovative association method that considers every detection box. Detection boxes with lower scores are processed by comparing their similarities with existing tracklets to accurately identify true objects while filtering out unwanted detections.

Within this context, the VegaEdge employs the ByteTrack algorithm, renowned for its efficient and robust tracking capabilities. ByteTrack's architecture uses deep association techniques, ensuring consistent tracking across frames, even for challenges like occlusions and complex interactions. Its performance ia shown in Table 4.1 on datasets like BDD100K and MOT20. Furthermore, ByteTrack boasts an impressive running speed without compromising on accuracy, making it an ideal choice for real-time applications such as VegaEdge. As discussed in [85], ByteTrack achieves metrics of 80.3 MOTA, 77.3 IDF1, and 63.1 HOTA on the MOT17 test set, all while maintaining a 30 FPS running speed.

4.5.3 Data Conditioning

The output from detection and tracking is rigorously cleaned to ensure precise input, as the quality of the input directly dictates the accuracy of trajectory forecasts in the next step. To obtain precise vehicular trajectories, we've performed targeted data filtration and smoothing:

1. Class-specific Inclusion: To eliminate potential noise from extraneous vehicular types, we selectively retain only cars, buses, and trucks.

2. Unidirectional Movement: To focus on the flow of incoming vehicles, we exclude the vehicles operating in non-targeted directions, thereby standardizing the directional flow and reducing complexity.

3. Temporal Presence Validation: Vehicles with transient appearances can introduce data anomalies. This validation process sets a minimum frame threshold, below which vehicular entries are deemed non-contributory and are subsequently removed.

4. Trajectory Continuity: Despite thorough validation in previous steps, some trajectories may have missing frames. We fill such gaps through interpolation techniques, ensuring continuous and complete trajectories.

In summary, the data cleansing and validation processes outlined above are crucial in ensuring the integrity and precision of vehicular trajectories used by VegaEdge's downstream tasks. By emphasizing class-specific inclusion, standardizing directional flow, validating temporal presence, and ensuring trajectory continuity, we lay the foundation for subsequent trajectory forecasts. This approach mitigates potential inaccuracies and fortifies our framework's reliability, positioning it to deliver consistent and high-quality results in real-world applications for which VegaEdge will be used.

4.5.4 Trajectory Prediction

We focus on highway-centric performance in the framework of VegaEdge's IoT applications. To meet this need, VegaEdge integrates the SotA PishguVe[25] trajectory prediction algorithm on highway datasets, ensuring fast and accurate results without straining the device.

PishguVe was selected for its ability to make predictions at the pixel level, as shown in Table 4.1, its proven track record of state-of-the-art accuracy on multiple datasets [2, 1, 25], and its efficient memory footprint. Table 4.1, CHD-HA and CHD-EL represents CHD-High Angle and CHD-Eye-level trajectories from CHD dataset respectively. Built on a fusion of graph isomorphism, convolutional neural networks, and attention mechanisms, PishguVe [25] can forecast future vehicle positions with a model size of only 133K parameters. The input to PishguVe is past trajectories of vehicles are represented by a set of absolute coordinates, denoted as P_v, and a set of relative coordinates, denoted as ΔP_v. The absolute coordinates are defined as $P_v = (x^t_{v,p}, y^t_{v,p})$, where $t = 1, ..., T_{in}$, $v \in 1, 2, ..., n$ representing the index of the vehicle and $x^t_{v,p}$ and $y^t_{v,p}$ are x and y coordinates of the center of bounding box of vehicle v at time t for past trajectory denoted by p. The predicted future trajectories are shown as $\hat{F}_v = (x^t_{v,f}, y^t_{v,f})$, here $t = (T_{in} + 1), ..., T_{pred}$, f denotes future trajectory, and $v \in 1, 2, ..., n$, are generated as a set of coordinates for each vehicle in the image.

4.5.5 Prediction-based Anomaly Detection

VegaEdge uses the trajectory prediction-based anomaly detection approach discussed in section 4.4 of the paper, utilizing ADE and angle-based anomaly detection techniques. These methods offer a straightforward and efficient approach to detecting anomalies. This streamlined process makes our proposed method well-suited for integration within VegaEdge's IoT-based framework, which operates on hardware and time-constrained embedded IoT platforms. This efficiency allows for quick anomaly

detection, enhancing the overall performance and responsiveness of the system. The performance of both approaches on two different datasets is demonstrated in the upcoming section.

4.5.6 VegaEdge Performance Benchmarking

VegaEdge's performance was evaluated across multiple platforms to understand its versatility and efficiency. Our testing platforms comprised a server with an Intel Xeon Silver 4114 CPU, Nvidia's V100 GPU, and Nvidia's Jetson Orin and Xavier NX boards. We chose the server setup as a reference point to contrast the performance metrics of the Jetson boards. These Jetson boards are designed for real-world tasks and are notable for their power efficiency, with Orin operating at 50W and Xavier NX at just 20W. Their efficiency and AI capabilities position them as ideal candidates for a wide range of IoT applications requiring edge computing.

Transitioning from the hardware evaluation, Table 4.1 shows the performance metrics of three algorithms VegaEdge utilizes for crafting its workflow, as shown in Algorithm 2 and Figure 4.2. In the domain of Object Detection, YOLOv8l achieves an mAP of 52.9 on COCO and 57.14 at mAP50 on BDD100K. It further scored 34.50 at mAP50-95. The Tracking algorithm, ByteTrack performs at a MOTA of 77.8 on the MOT20 dataset. Lastly, in trajectory prediction, the PishguVe algorithm was assessed on three distinctive datasets. It registered a Pixels/ADE of 20.75 and 16.81 on

Table 4.1: Accuracy comparison of each algorithm.

Task	**Method**	**Performance**	**Dataset**
Object Detection	YOLOv8l [113]	52.9 (mAP)	COCO [115]
		57.14 (mAP)@mAP50	BDD100K [116]
		34.50 (mAP)@mAP50-95	BDD100K [116]
Tracking	ByteTrack [85]	77.8 (MOTA)	MOT20 [117]
Path Prediction	PishguVe [25]	20.75 (Pixels/ADE)	CHD-EL [25]
		16.81 (Pixels/ADE)	CHD-HA [25]
		0.77(m/ADE)	NGSIM [2, 1]

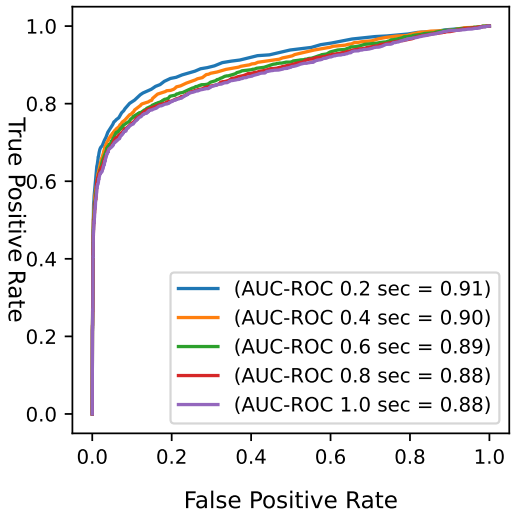

Figure 4.3: Fine grain (0.2s step)

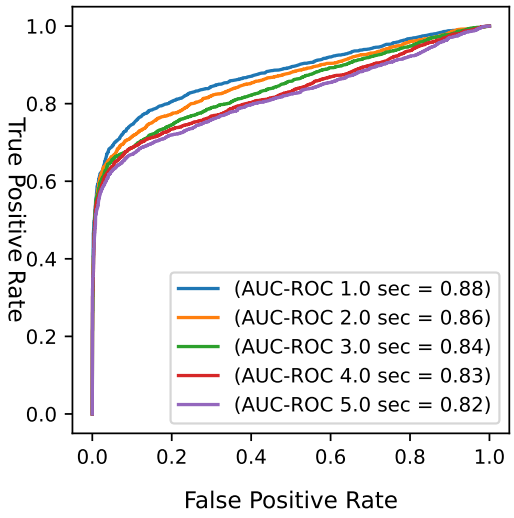

Figure 4.4: Coarse grain (1s step)

Figure 4.5: AUC-ROC with varying detection windows for ADE-based anomaly detection method on NGSIM dataset. (a) Shows plot from 0s to 1s with time-step of 0.2 s (b) Shows plot from 1s to 5s with time-step of 1.0 s

CHD-EL and CHD-HA, respectively, and a commendable m/ADE of 0.77 on NGSIM.

4.6 Results

4.6.1 Anomaly detection on NGSIM Dataset:

In this section, we evaluate our anomaly detection methodology on adversarial trajectories. These trajectories are derived using the NGSIM dataset with bird's eye camera-view of the trajectories, following the adversarial attack approach [112], adopted by [109]. Our evaluation encompasses both the ADE and angle-based anomaly detection techniques. Our tests include both adversarial generated trajectories [112] and actual real-world data from the NGSIM test dataset, similar to the study in [118].

The ADE-based anomaly detection method study revealed a distinct pattern regarding the Area Under the Receiver Operating Characteristic curve (AUC-ROC). The ADE-based anomaly detection method consistently performed best at an AUC-ROC of 0.91 for an ADE window of 0.2s, as visualized in Figs. 4.5 and 4.3. The angle-based approach initially outperformed the ADE-based method for short pre-

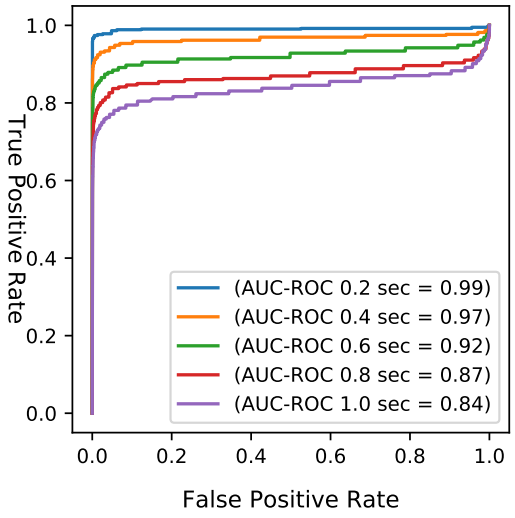

Figure 4.6: Fine grain (0.2s step)

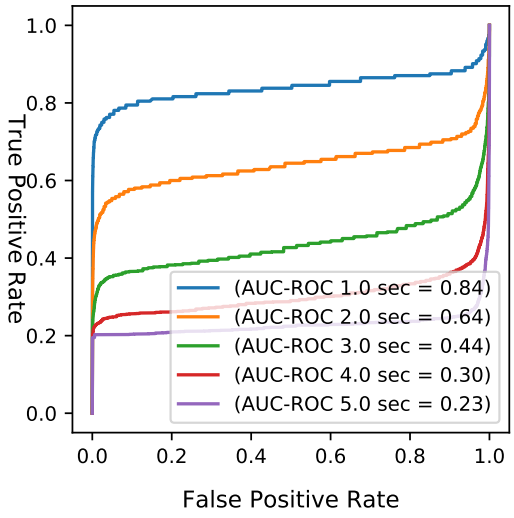

Figure 4.7: Coarse grain (1s step)

Figure 4.8: AUC-ROC with varying detection windows for angle-based anomaly detection method on NGSIM dataset. (a) Shows plot from 0s to 1s with time-step of 0.2 s (b) Shows plot from 1s to 5s with time-step of 1.0 s

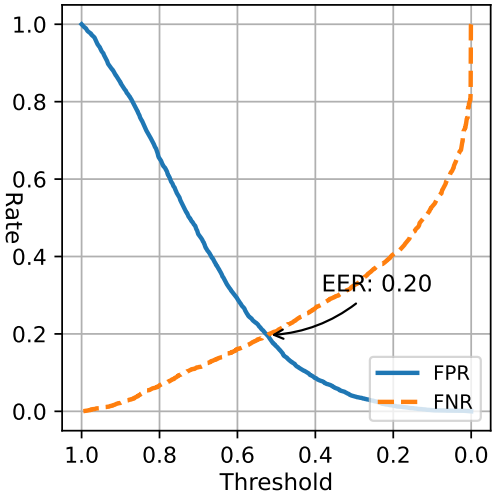

Figure 4.9: ADE-based

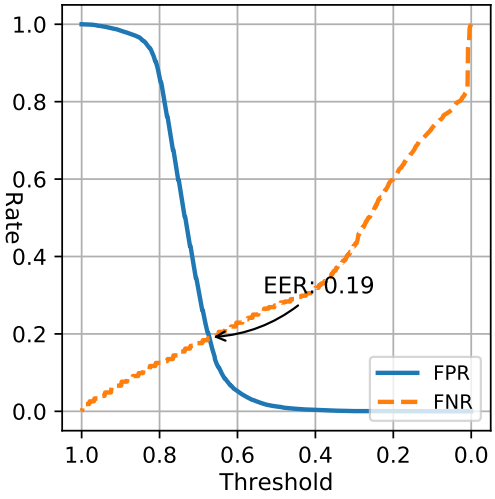

Figure 4.10: Angle-based

Figure 4.11: EER plots for anomaly detections with 1s trajectory window for NGSIM dataset adversarial trajectories: (a) ADE-based anomaly, and (b) Angle-based anomaly.

Table 4.2: EER for various Time Thresholds using ADE-and Angle based Anomaly on NGSIM Adversarial trajectories

Time-step (s)	**EER**	
	ADE-based	Angle-based
0.2	0.16	0.02
0.4	0.17	0.05
0.6	0.19	0.10
0.8	0.19	0.15
1.0	0.20	0.19
2.0	0.22	0.38
3.0	0.24	0.56
4.0	0.25	0.69
5.0	0.26	0.77

diction windows, as seen in Fig. 4.6, but its efficacy declined with larger windows, evident in Fig. 4.7. Such behavior in the angle-based approach may stem from how anomalies are generated by applying constrained perturbations to real-world trajectories, making anomalies challenging to discern over more extended prediction periods.

The Equal Error Rate (EER) plot in Fig. 4.11 shows the EER value obtained by plotting False Negative and False Positive Rate for ADE and Angle anomaly methods for 1 second of predicted trajectory. Table 4.2 presents the EER for two anomaly detection methods on NGSIM adversarial trajectories across different time-step thresholds. The Angle method outperforms the ADE approach at shorter thresholds, such as 0.2s and 0.4s. However, as the time threshold grows, their performance converges. By 5.0 seconds, the EERs are 0.26 for ADE and 0.77 for the Angle method, indicating a faster performance drop for the Angle approach over extended time steps.

4.6.2 Anomaly detection on CAD

In this section, we evaluate our anomaly detection approach on real-world trajectories sourced from CAD consisting of high-angle (CHD-HA) and eye-level (CHD-EL) camera-view of highway vehicles, as introduced in section 4.3. To offer a comprehensive view of the results, the AUC-ROC values for both fine and coarse grain time

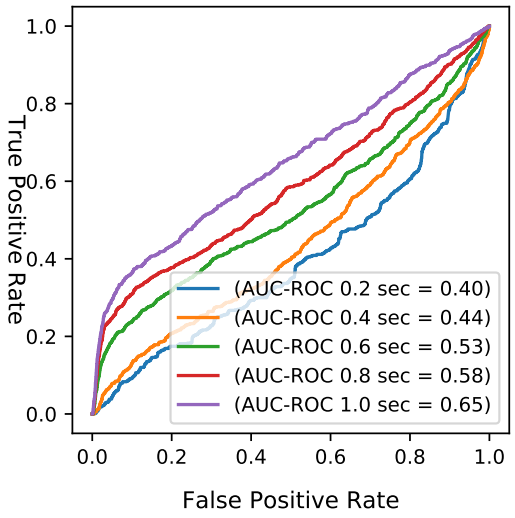

Figure 4.12: Fine grain (0.2s)

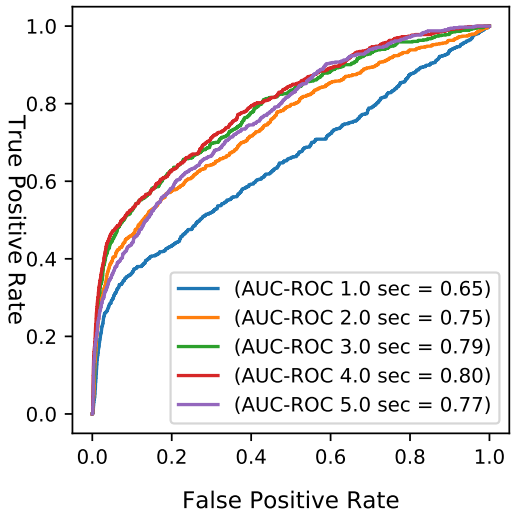

Figure 4.13: Coarse grain (1s)

Figure 4.14: AUC-ROC with varying prediction windows for ADE-based anomaly detection method on CAD. (a) Shows plot from 0s to 1s with a time-step of 0.2 s. (b) Shows plot from 1s to 5s with time-step of 1.0 s.

steps are graphically represented in Fig. 4.14 for the ADE-based anomaly detection method and in Fig. 4.17 for the angle-based method.

Expanding on the earlier analysis, Table 4.3 provides a breakdown of the Equal Error Rate (EER) performance for various time thresholds, contrasting the ADE and angle-based anomaly detection methods on CAD's data. It is clear that the Angle approach consistently outperforms the ADE method across all examined time steps. Starting from an EER of 0.48 at the 0.2s mark, the angle method demonstrates a steady improvement, with an EER of 0.12 at the 5s threshold. In contrast, the ADE-based method initiates with an EER of 0.58 at 0.2s and gradually improves to 0.32 at 5s. These metrics show the superior efficacy of the angle-based approach, especially since the prediction windows are elongated, making it a more robust choice for anomaly detection in the context of the CAD.

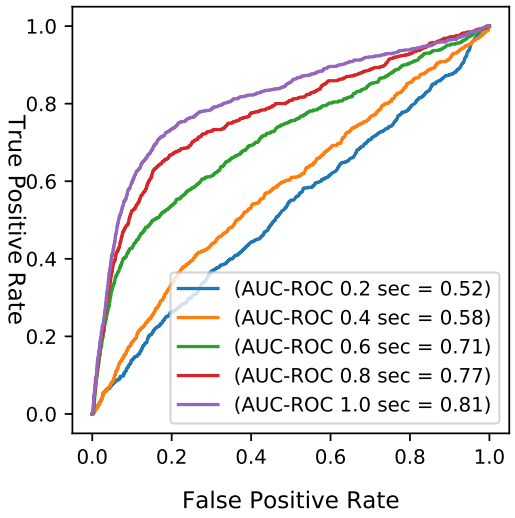

Figure 4.15: Fine grain (0.2s)

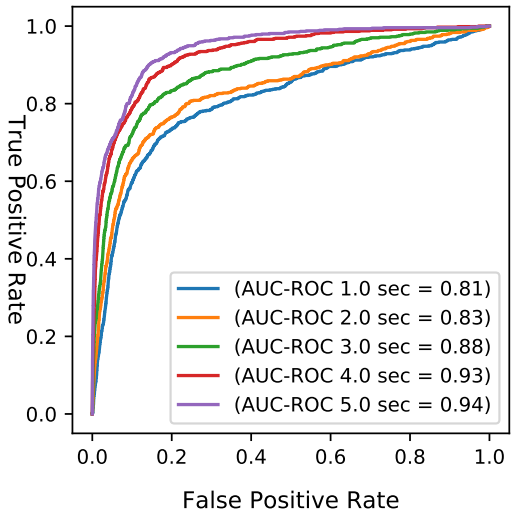

Figure 4.16: Coarse grain (1s)

Figure 4.17: AUC-ROC with varying prediction windows for Angle-based anomaly detection method on CAD. (a) Shows plot from 0s to 1s with a time-step of 0.2 s. (b) Shows plot from 1s to 5s with time-step of 1.0 s.

4.6.3 Real-world vs Adversarial Anomaly Trajectories

Intriguingly, the real-world trajectories demonstrate an inverse trend compared to the results of the adversarial anomaly dataset. Specifically, the AUC-ROC values for the ADE anomaly detection method peak at a 4 and 5-second prediction window, recording the area of 0.80 for a 4s window as shown in Fig. 4.13. This suggests a higher sensitivity of the ADE method to longer prediction windows when applied to real-world data. Similarly, in Fig. 4.16 the angle-based method exhibits stellar performance with an AUC-ROC of 0.94 for the same 5s window. Such observations indicate that while synthetic or constrained trajectory datasets may favor short prediction windows, real-world trajectories might inherently contain more distinguishable anomalies in longer prediction intervals.

Comparing the EER values from the real-world CAD trajectories (Table 4.3) with those from the adversarial NGSIM trajectories (Table 4.2), several striking differences emerge. The NGSIM adversarial trajectories exhibit substantially lower EER values

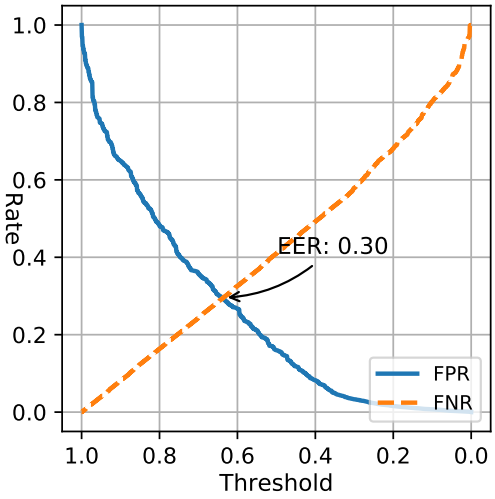

Figure 4.18: ADE-based

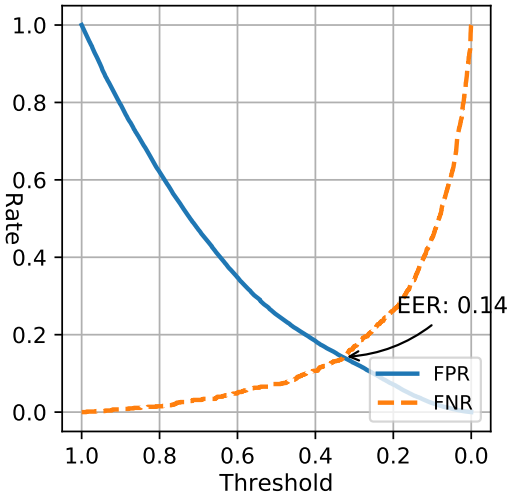

Figure 4.19: Angle-based

Figure 4.20: EER plots for 4s anomaly detections for CAD dataset: (a) ADE-based anomaly, and (b) Angle-based anomaly.

in the initial time steps, especially for the angle-based approach. For example, at the 0.2-second mark, the NGSIM data record a notably lower EER of 0.02 for the angle-based method than 0.48 for the CAD dataset.

A potential reason for this marked divergence might be the inherent nature of the datasets. The NGSIM adversarial trajectories, being synthetically generated, likely present more pronounced and discernible anomalies that the detection methods can more readily identify, especially within shorter prediction windows. In contrast, the CAD real-world trajectories, a genuine reflection of real-world driving behaviors, might be embedded with subtler and more intricate anomalies. These nuances could pose more significant challenges in detection, resulting in higher EER values, especially in the shorter time-step intervals. Moreover, the complexities and variances found in real-world driving behaviors could introduce a wider array of anomalies, making distinguishing between normal and anomalous patterns more intricate for the detection methods when applied to the CAD.

Table 4.3: EER for various Time Thresholds using ADE and Angle based Anomaly on CAD

Time-step (s)	**EER**	
	ADE-based	**Angle-based**
0.2	0.58	0.48
0.4	0.55	0.43
0.6	0.50	0.35
0.8	0.46	0.28
1.0	0.41	0.24
2.0	0.34	0.22
3.0	0.30	0.17
4.0	0.30	0.14
5.0	0.32	0.12

4.6.4 VegaEdge Performance

In our evaluation, we primarily focus on the performance of VegaEdge on the Jetson platforms. With its superior computational capabilities, the server platform serves as a benchmarking reference to underscore the efficiency and feasibility of deploying VegaEdge in more constrained environments.

4.6.4.1 Performance on Jetson Platforms

Table 4.4 delineates the throughput of VegaEdge across different road scenarios and traffic densities. On Jetson platforms, VegaEdge's performance showcases its adaptability and efficiency, particularly given the embedded nature of these devices.

For the *3 lanes high traffic density* scenario, the Jetson Orin processes 758 trajectories every second, with 140 unique vehicles detected per second and the intricate nature of merging scenarios. This throughput ensures real-time processing capabilities essential for many applications. Meanwhile, the Xaview NX, while trailing with 243 trajectories per second, still provides a usable metric for less time-sensitive tasks or preliminary data-gathering efforts.

The *2 lanes with workzone with low traffic density* scenario provides a different

Table 4.4: Throughput (processed trajectories per second) comparison of VegaEdge on the three platforms across different road types and traffic densities (expressed as vehicle trajectories processed per second)

Road Type (Traffic Density)	**Throughput**		
	Server	**Jetson Orin**	**Xaview NX**
3 lanes and merger (140)	1770	758	243
2 lanes with workzone (18)	2868	132	47
2 lanes (simulated video) (13)	1050	92	31

challenge, simulating a common urban environment with traffic modulations due to work zones. In this context, the Jetson Orin delivers a performance of 132 trajectories per second, making it ideal for surveillance and alert systems in smart city setups. This reduced throughput is attributed to the fewer vehicles present in the scene, not the capability constraints. Meanwhile, the Xavier NX offers 47 trajectories per second, which may be apt for tasks requiring less frequent monitoring.

4.6.4.2 Digital Twin Systems

Another aspect of our evaluation is the *2 lanes (simulated video) (13)* scenario. The Jetson Orin's capability to process 92 trajectories per second in a simulated environment underscores its potential utility in digital twin systems. Digital twin systems, which mirror and simulate real-world environments, are important in predictive maintenance, system optimization, and various simulation-driven tasks. The ability of VegaEdge to run efficiently on simulated data on the Jetson Orin emphasizes its versatility and readiness for such advanced applications.

4.6.5 Highway Workzone Safety Application

In highway work zones, safeguarding workers from oncoming vehicles is a primary concern. Through trajectory prediction combined with anomaly detection, VegaEdge detects trajectories that may pose threats and alerts workers. To achieve this objective, the proposed design must demonstrate real-time performance on edge devices.

Table 4.5: Comparison of Anomaly detection Prediction Windows, Buffer Times (Excluding 3s Input Trajectory), and Vehicle Distances from Camera at 60 mph.

Error Detection Window (s)	**Buffer Time (s)**	**Distance (m)**
0.2	8.8	235
1.0	8	213
3.0	6	160
5.0	4	107

The efficacy of VegaEdge, particularly when implemented on Jetson Orin, is exemplified in Table 4.5.

Given an error detection window of 0.2s, VegaEdge provides a buffer time of 8.8s, corresponding to a 235m distance from a camera for a vehicle moving at 60 mph. As the detection window widens, the buffer diminishes but remains noteworthy, with 1s, 3s, and 5s windows yielding buffer times of 8s, 6s, and 4s, respectively. A clear trade-off emerges between larger buffer windows and heightened accuracy, as noted in the anomaly detection results for CAD in section 4.6.2.

In hazardous highway work zones, these buffer times are beneficial and vital. Even minor increments in warning time can significantly alter outcomes. VegaEdge's ability to grant these buffers, particularly on Jetson platforms, underscores its practicality in real-world scenarios and its role in bolstering worker safety.

Table 4.6: Comparison of VegaEdge Power consumption and Throughput ((Trajectories processed per second) on Jetson Orin Power Modes for real word highway ($\sim$ 140 Vehicles detected per second across 30 frames.). Power is measured in Watts.

Power Mode	**Total Power**	**GPU Power**	**CPU Power**	**Throughput**
MAXN	18.14	3.66	8.80	758
30W	11.44	3.09	3.72	477
15W	8.43	2.82	1.3	214

4.6.6 VegaEdge Power Analysis on IoT Platform

In this subsection, we conduct a comprehensive power consumption analysis of the Nvidia Jetson Orin platform with the primary objective of assessing the practical utility and performance of VegaEdge across various power modes. We report the total power consumed across all the channels (P_{Total}) calculated using the following equation:

$$P_{\text{Total}} = P_{\text{GPU}} + P_{\text{CPU}} + P_{\text{IOs}} + P_{\text{AO}} \tag{4.4}$$

Where P_{GPU} is the total power consumed by the GPU and SOC cores, P_{CPU} is the power consumed by the CPU and CV cores, and P_{IOs} is the power consumed by the system's 5V rail for various input and output ports, respectively on one of the power monitors [119]. P_{AO} stands for power consumed by Always On (AO) power rail on another power monitor [119].

In Table 4.6, VegaEdge's power consumption and throughput on the Jetson Orin are evaluated across different power modes for a real-world highway scenario. At the MAXN (40W) setting, the system processes 758 trajectories per second, consuming 18.14W, and reducing the power mode to 30W and 15W results in decreased throughputs of 477 and 214 trajectories per second, respectively, with corresponding reductions in power consumption. Despite the higher power demands compared to typical IoT devices, VegaEdge on Jetson Orin showcases a valuable trade-off between power consumption and high-throughput processing, making it a viable solution for edge applications requiring rapid data processing.

4.7 Conclusion

In this work introduced a minimalist anomaly detection approach based on predicted trajectories and showcased its effectiveness across various prediction windows on adversarial and real-world anomalies. We presented VegaEdge, a real-time high-

way safety solution optimized IoT-edge applications utilizing our anomaly detection method, achieving a throughput of up to 758 processed vehicle trajectories per second in high-traffic conditions. Furthermore, our application of VegaEdge demonstrated its ability to adapt buffer times for workzone personnel, highlighting the trade-off between buffer time and accuracy for such applications. The introduction of the Carolinas Anomaly Dataset (CAD) as a dedicated resource for real-world highway anomaly detection, combined with our innovative approach, highlights the potential of IoT and AI in advancing highway safety.

CHAPTER 5: CONCLUSIONS

In this dissertation, we presented a series of pivotal contributions to the intersection of artificial intelligence and edge-centric intelligent transportation applications. These contributions, namely DeepTrack, CHD, CAD, PishguVe, and VegaEdge, showcase the field's breadth and depth of advancements. Accompanying these is a streamlined anomaly detection method optimized for real-time operations, leveraging trajectory prediction. These elements together represent a big change, making AI-powered transportation systems even safer and more efficient on highways.

The initial contribution of this work DeepTrack emerged as an example of efficient deep learning. Designed for efficiency in the domain of deep learning, DeepTrack doesn't merely maintain commendable accuracyâit redefines it, with a marked reduction in model size and computational complexity. This deems it suitable for embedded edge systems. The decision to pivot from traditional LSTM units to ATCN for encoding vehicle dynamics, particularly through depthwise convolution, has shown to be both innovative and effective. This move offers the dual advantage of minimizing the model's footprint and its operational intricacies. Furthermore, our rigorous evaluations showcase DeepTrack's superiority, particularly when benchmarked against the CS-LSTM and CF-LSTM on the NGSIM dataset. It surpassed the then-leading CS-LSTM, reducing prediction errors by 9.09% and 11.56% and concurrently trimming the model's operations and size by 10.49% and 18.5% on the NGSIM dataset. In a head-to-head comparison with CF-LSTM using ADE as a metric, DeepTrack mirrored its performance but achieved a noteworthy reduction in both its operational footprint and model size, by 10.49% and 18.5% respectively. Its promise in both enhancing performance and optimizing resource utilization sets a precedent for subsequent models

and research in the field.

Following our exploration of DeepTrack, another cornerstone of this dissertation is the introduction of the Carolinas Highway dataset (CHD). This dataset, assembled from over 1.6M high-resolution images, encapsulates more than 338,000 vehicle trajectories sourced from eight distinct highway locations, providing varied perspectives. It stands as a unique benchmark, vital for gauging the efficacy of trajectory prediction models and enhancing various applications within Intelligent Transportation Systems (ITS).

Furthermore, in pursuit of higher prediction accuracy as it is directly related to safety in critical applications, the research introduced PishguVe. Harnessing graph-based attention and an attentive neural network, PishguVe distills critical features from complex traffic scenarios, delivering real-time results. Constructing graphs of vehicles within a scene captures the interdependencies and behaviors characteristic of diverse driving conditions. Empirical evaluations, particularly on the NGSIM and CHD datasets, confirm PishguVe's superiority. Not only does it set a new benchmark for vehicle trajectory prediction across multiple perspectives, but it also is betters than other SotA models. For instance, on the NGSIM dataset, PishguVe registers improvements of 12.50% in ADE and 10.20% in FDE over the existing state-of-the-art. Similarly, its performance on the CHD reaffirms its prowess, with marked enhancements in ADE and FDE metrics.

Concluding our narrative on the advancements within this dissertation, we delve into the realm of anomaly detection. We introduce the Carolinas Anomaly Dataset (CAD) crafted from CHD, is a collection of highway-based videos with atypical driving behaviors for real-world highway anomaly detection. Our research also presents a streamlined anomaly detection paradigm, grounded in predicted trajectories. Its efficacy stands tested across a number of prediction windows, handling both adversarial and real-world anomalies. Further elevating the discourse is VegaEdge, a dedicated

real-time highway safety mechanism tailored for IoT-edge applications. By harnessing the power of our proposed anomaly detection technique, VegaEdge can process 758 vehicle trajectories per second under high-traffic scenarios. It showcases the need to balance the buffer times for work zone personnel while balancing prediction accuracy. The confluence of our pioneering anomaly detection methodology with other vision-based detection, tracking and prediction algorithms underscores the transformative potential of IoT and AI in sculpting the future of highway safety.

As this dissertation concludes, it's evident that the collective contributions lay a solid foundation and direction for researchers, practitioners, and policymakers in the rapidly evolving landscape of artificially intelligent transportation systems.

REFERENCES

[1] J. Colyar and J. Halkias, "Next generation simulation (NGSIM), Interstate 80 freeway dataset. FHWA-HRT-06-137," 2006.

[2] J. Colyar and J. Halkias, "Next generation simulation (NGSIM), US Highway-101 dataset. FHWA-HRT-07-030.," 2007.

[3] "Fatality Analysis Reporting System (FARS), Motor vehicle traffic crashes (1994-2019)," 2019.

[4] "Fatality Analysis Reporting System (FARS), Vehicles involved in fatal crashes," 2019.

[5] N. H. T. S. Administration, "Early estimate of 2021 traffic fatalities," 2022.

[6] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2118–2125, 2018.

[7] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," *arXiv preprint arXiv:2301.00493*, 2023.

[8] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," in *Conference on Robot Learning*, pp. 409–418, PMLR, 2021.

[9] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, 2020.

[10] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4604–4612, 2020.

[11] Y. Wang, S. Zhao, R. Zhang, X. Cheng, and L. Yang, "Multi-vehicle collaborative learning for trajectory prediction with spatio-temporal tensor fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 236–248, 2020.

[12] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9554–9567, 2022.

[13] Y. Cai, L. Dai, H. Wang, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "Pedestrian motion trajectory prediction in intelligent driving from far shot first-person perspective video," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5298–5313, 2021.

[14] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 33–47, 2020.

[15] L. Hou, L. Xin, S. E. Li, B. Cheng, and W. Wang, "Interactive trajectory prediction of surrounding road users for autonomous driving using structural-lstm network," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4615–4625, 2020.

[16] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 1468–1476, IEEE Computer Society, 2018.

[17] X. Ren, D. Wang, M. Laskey, and K. Goldberg, "Learning traffic behaviors by extracting vehicle trajectories from online video streams," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1276–1283, 2018.

[18] D. Tian, C. Zhang, X. Duan, and X. Wang, "An automatic car accident detection method based on cooperative vehicle infrastructure systems," *IEEE Access*, vol. 7, pp. 127453–127463, 2019.

[19] D. Singh and C. K. Mohan, "Deep spatio-temporal representation for detection of road accidents using stacked autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 879–887, 2019.

[20] J. Sochor, R. JurÃ¡nek, J. Å paÅhel, L. MarÅ¡Ãk, A. Å irokÃœ, A. Herout, and P. ZemÄÃk, "Comprehensive data set for automatic single camera visual speed measurement," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1633–1643, 2019.

[21] A. P. Shah, J.-B. Lamare, T. Nguyen-Anh, and A. Hauptmann, "Cadp: A novel dataset for cctv traffic camera based accident analysis," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–9, IEEE, 2018.

[22] C. Nnaji, J. Gambatese, and H. W. Lee, "Work zone intrusion: Technology to reduce injuries and fatalities," *Professional Safety*, vol. 63, no. 04, pp. 36–41, 2018.

[23] F. H. Administration, "Work zones are a sign to slow down," 2022.

[24] Y. Hu, Y. Zhang, Y. Wang, and D. Work, "Detecting socially abnormal highway driving behaviors via recurrent graph attention networks," in *Proceedings of the ACM Web Conference 2023*, pp. 3086–3097, 2023.

[25] V. Katariya, G. A. Noghre, A. D. Pazho, and H. Tabkhi, "A pov-based highway vehicle trajectory dataset and prediction architecture," *arXiv preprint arXiv:2303.06202*, 2023.

[26] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.

[27] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 14062–14071, IEEE, 2020.

[28] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using lstms with spatial-temporal attention mechanisms," *IEEE Intelligent Transportation Systems Magazine*, pp. 0–0, 2021.

[29] V. D. H. Richard and H. Jeroen, "Time-to-Collision and Collision avoidance systems," *International Co-operation on Theories and Concepts in Traffic safety (ICTCT)*, 1994.

[30] L. Lin, W. Li, and S. Peeta, "Efficient data collection and accurate travel time estimation in a connected vehicle environment via real-time compressive sensing," *Journal of Big Data Analytics in Transportation*, vol. 1, 12 2019.

[31] X. Xie, C. Zhang, Y. Zhu, Y. N. Wu, and S. Zhu, "Congestion-aware multi-agent trajectory prediction for collision avoidance," *CoRR*, vol. abs/2103.14231, 2021.

[32] E. Kafer, C. Hermes, C. Wöhler, H. J. Ritter, and F. Kummert, "Recognition of situation classes at road intersections," in *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*, pp. 3960–3965, IEEE, 2010.

[33] A. Lawitzky, D. Althoff, C. F. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss, "Interactive scene prediction for automotive applications," in *2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast City, Australia, June 23-26, 2013*, pp. 1028–1033, IEEE, 2013.

[34] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), June 17-19, 1997, San Juan, Puerto Rico*, pp. 994–999, IEEE Computer Society, 1997.

[35] W. Kun, W. Shaobo, Z. Pan, Y. Biao, H. Weixin, and L. Huawei, "Vehicle trajectory prediction by knowledge-driven lstm network in urban environments," *Journal of Advanced Transportation*, 2020.

[36] L. Xin, P. Wang, C. Chan, J. Chen, S. E. Li, and B. Cheng, "Intention-aware long horizon trajectory prediction of surrounding vehicles using dual LSTM networks," in *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018* (W. Zhang, A. M. Bayen, J. J. S. Medina, and M. J. Barth, eds.), pp. 1441–1446, IEEE, 2018.

[37] J. Mercat, T. Gilles, N. E. Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-head attention for multi-modal joint vehicle motion forecasting," in *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pp. 9638–9644, IEEE, 2020.

[38] M. Mendieta and H. Tabkhi, "CARPe posterum: A convolutional approach for real-time pedestrian path prediction," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 2346–2354, AAAI Press, 2021.

[39] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[40] M. Abdel-Aty, R. J. Cunningham, V. V. Gayah, and L. Hsia, "Dynamic variable speed limit strategies for real-time crash risk reduction on freeways," *Transportation Research Record*, vol. 2078, no. 1, pp. 108–116, 2008.

[41] N. Dutta, R. A. Boateng, and M. D. Fontaine, "Safety and operational effects of the interstate 66 active traffic management system," *Journal of Transportation Engineering, Part A: Systems*, vol. 145, no. 3, p. 04018089, 2019.

[42] H. Yang, K. Ozbay, and K. Xie, "Assessing the risk of secondary crashes on highways," *Journal of Safety Research*, vol. 49, pp. 143.e1–149, 2014. Proceedings of the International Conference on Road Safety (RSS2013).

[43] R. A. Raub and J. L. Schofer, "Managing incidents on urban arterial roadways," *Transportation Research Record*, vol. 1603, no. 1, pp. 12–19, 1997.

[44] A. Kesting and M. Treiber, "How reaction time, update time, and adaptation time influence the stability of traffic flow," *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, no. 2, pp. 125–137, 2008.

[45] L. Yu, A. R. Bill, M. V. Chitturi, and D. A. Noyce, "On-duty struck-by crashes: Characteristics and contributing factors," *Transportation Research Record*, vol. 2386, no. 1, pp. 112–120, 2013.

[46] L. Tijerina, M. Shulman, and D. Kochhar, "Committee report: Conspicuity enhancement for police interceptor rear-end crash mitigation," *Ford Motor Company, Dearborn*, 2003.

[47] R. T. CODD III, "Fatal crash investigation: First responder issues," in *Behind the Badge*, pp. 261–277, Routledge, 2014.

[48] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[49] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *CoRR*, vol. abs/1611.02344, 2016.

[50] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *CoRR*, vol. abs/1705.03122, 2017.

[51] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, IEEE Computer Society, 2016.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456, JMLR.org, 2015.

[54] P. Kothari, S. Kreiss, and A. Alahi, "Human trajectory forecasting in crowds: A deep learning perspective," *CoRR*, vol. abs/2007.03639, 2020.

[55] J. Yan, Z. Peng, H. Yin, J. Wang, X. Wang, Y. Shen, W. Stechele, and D. Cremers, "Trajectory prediction for intelligent vehicles using spatial-attention mechanism," *IET Intelligent Transport Systems*, vol. 14, no. 13, pp. 1855–1863, 2020.

[56] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[57] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las*

Vegas, NV, USA, June 27-30, 2016, pp. 961–971, IEEE Computer Society, 2016.

[58] B. Zhang and J. Zhang, "A traffic surveillance system for obtaining comprehensive information of the passing vehicles based on instance segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7040–7055, 2021.

[59] B. Olofsson and L. Nielsen, "Using crash databases to predict effectiveness of new autonomous vehicle maneuvers for lane-departure injury reduction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3479–3490, 2021.

[60] F. H. Administration, "Fhwa work zone facts and statistics." `https://ops.fhwa.dot.gov/wz/resources/facts_stats.htm`, 2023. Accessed: February 15, 2023.

[61] A. G. C. of America, "Survey: Work zone crashes up 4% in 2022," 2022. Accessed: February 15, 2023.

[62] R. Ke, Z. Li, S. Kim, J. Ash, Z. Cui, and Y. Wang, "Real-time bidirectional traffic flow parameter estimation from aerial videos," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 890–901, 2017.

[63] S. Sabeti, O. Shoghli, M. Baharani, and H. Tabkhi, "Toward ai-enabled augmented reality to enhance the safety of highway work zones: Feasibility, requirements, and challenges," *Advanced Engineering Informatics*, vol. 50, p. 101429, 2021.

[64] X. Huang, P. He, A. Rangarajan, and S. Ranka, "Intelligent intersection: Two-stream convolutional networks for real-time near-accident detection in traffic video," *ACM Trans. Spatial Algorithms Syst.*, vol. 6, no. 2, pp. 10:1–10:28, 2020.

[65] A. Clausse, S. Benslimane, and A. de La Fortelle, "Large-scale extraction of accurate vehicle trajectories for driving behavior learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2391–2396, 2019.

[66] C. Wang, A. Musaev, P. Sheinidashtegol, and T. Atkison, "Towards detection of abnormal vehicle behavior using traffic cameras," in *Big Data–BigData 2019: 8th International Congress, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 8*, pp. 125–136, Springer, 2019.

[67] T. Ren, Y. Xie, and L. Jiang, "Cooperative highway work zone merge control based on reinforcement learning in a connected and automated environment," *Transportation research record*, vol. 2674, no. 10, pp. 363–374, 2020.

[68] G. A. Noghre, V. Katariya, A. D. Pazho, C. Neff, and H. Tabkhi, "Pishgu: Universal path prediction architecture through graph isomorphism and attentive convolution," *arXiv preprint arXiv:2210.08057*, 2022.

[69] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 6120–6127, 2019.

[70] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. O. noro Rubio, "Extremely overlapping vehicle counting," in *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2015.

[71] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[72] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, 2020.

[73] S. Lyu, M.-C. Chang, D. Du, W. Li, Y. Wei, M. Del Coco, P. Carcagnì, A. Schumann, B. Munjal, D.-H. Choi, *et al.*, "Ua-detrac 2018: Report of avss2018 & iwt4s challenge on advanced traffic monitoring," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, IEEE, 2018.

[74] S. Lyu, M.-C. Chang, D. Du, L. Wen, H. Qi, Y. Li, Y. Wei, L. Ke, T. Hu, M. Del Coco, *et al.*, "Ua-detrac 2017: Report of avss2017 & iwt4s challenge on advanced traffic monitoring," in *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pp. 1–7, IEEE, 2017.

[75] X. Ren, D. Wang, M. Laskey, and K. Goldberg, "Learning traffic behaviors by extracting vehicle trajectories from online video streams," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1276–1283, IEEE, 2018.

[76] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation: A set of best practices for high quality, economical video labeling," *International journal of computer vision*, vol. 101, pp. 184–204, 2013.

[77] Tzutalin, "Heartexlabs/labelimg: Labelimg."

[78] X. Li, X. Ying, and M. C. Chuah, "Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving," *arXiv preprint arXiv:1907.07792*, 2019.

[79] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using lstms with spatial–temporal attention mechanisms," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 197–208, 2021.

[80] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pp. 683–700, Springer, 2020.

[81] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14424–14432, 2020.

[82] A. D. Pazho, G. A. Noghre, B. R. Ardabili, C. Neff, and H. Tabkhi, "Chad: Charlotte anomaly dataset," *arXiv preprint arXiv:2212.09258*, 2022.

[83] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, and P. Rai, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020.

[84] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 2633–2642, Computer Vision Foundation / IEEE, 2020.

[85] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," 2022.

[86] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, 2020.

[87] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2118–2125, 2018.

[88] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[89] V. Katariya, M. Baharani, N. Morris, O. Shoghli, and H. Tabkhi, "Deeptrack: Lightweight deep learning for vehicle trajectory prediction in highways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18927–18936, 2022.

[90] F. H. Administration, "Fhwa work zone facts and statistics," 2023. Accessed: [Insert Date of Access].

[91] C. Nnaji, J. Gambatese, H. W. Lee, and F. Zhang, "Improving construction work zone safety using technology: A systematic review of applicable technologies," *Journal of traffic and transportation engineering (English edition)*, vol. 7, no. 1, pp. 61–75, 2020.

[92] Y. Tian, *Earth observation data for assessing global urbanization-sustainability nexuses*. PhD thesis, Wageningen University, 2023.

[93] R. Izquierdo, Á. Quintanar, D. F. Llorca, I. G. Daza, N. Hernández, I. Parra, and M. Á. Sotelo, "Vehicle trajectory prediction on highways using bird eye view representations and deep learning," *Applied Intelligence*, vol. 53, no. 7, pp. 8370–8388, 2023.

[94] V. Radermecker and L. Vanhaverbeke, "Estimation of public charging demand using cellphone data and points of interest-based segmentation," *World Electric Vehicle Journal*, 2023.

[95] S. Chen, L. Piao, X. Zang, Q. Luo, J. Li, J. Yang, and J. Rong, "Analyzing differences of highway lane-changing behavior using vehicle trajectory data," *Physica A Statistical Mechanics and its Applications*, vol. 624, p. 128980, Aug. 2023.

[96] K. Deng, "Anomaly detection of highway vehicle trajectory under the internet of things converged with 5g technology," *Complexity*, vol. 2021, pp. 1–12, 2021.

[97] J. Wu, X. Wang, X. Xiao, and Y. Wang, "Box-level tube tracking and refinement for vehicles anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4112–4118, 2021.

[98] Y. Li, J. Wu, X. Bai, X. Yang, X. Tan, G. Li, S. Wen, H. Zhang, and E. Ding, "Multi-granularity tracking with modularlized components for unsupervised vehicles anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 586–587, 2020.

[99] A. Danesh Pazho, C. Neff, G. A. Noghre, B. R. Ardabili, S. Yao, M. Baharani, and H. Tabkhi, "Ancilia: Scalable intelligent video surveillance for the artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 14940–14951, 2023.

[100] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014.

[101] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2020.

[102] R. Chandra, U. Bhattacharya, C. Roncal, A. Bera, and D. Manocha, "Robusttp: End-to-end trajectory prediction for heterogeneous road-agents in dense traffic with noisy sensor inputs," in *Proceedings of the 3rd ACM Computer Science in Cars Symposium*, pp. 1–9, 2019.

[103] C. Liu, D. Q. Huynh, Y. Sun, M. Reynolds, and S. Atkinson, "A vision-based pipeline for vehicle counting, speed estimation, and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7547–7560, 2020.

[104] P. Wei, H. Shi, J. Yang, J. Qian, Y. Ji, and X. Jiang, "City-scale vehicle tracking and traffic flow estimation using low frame-rate traffic cameras," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pp. 602–610, 2019.

[105] N. Balamuralidhar, S. Tilon, and F. Nex, "Multeye: Monitoring system for real-time vehicle detection, tracking and speed estimation from uav imagery on edge-computing platforms," *Remote sensing*, vol. 13, no. 4, p. 573, 2021.

[106] N. Peri, P. Khorramshahi, S. S. Rambhatla, V. Shenoy, S. Rawat, J.-C. Chen, and R. Chellappa, "Towards real-time systems for vehicle re-identification, multi-camera tracking, and anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 622–623, 2020.

[107] S. Tak, J.-D. Lee, J. Song, and S. Kim, "Development of ai-based vehicle detection and tracking system for c-its application," *Journal of advanced transportation*, vol. 2021, pp. 1–15, 2021.

[108] F. Jiang, S. A. Tsaftaris, Y. Wu, and A. K. Katsaggelos, "Detecting anomalous trajectories from highway traffic data," *Electrical Engineering and Computer Science Department, Northwestern University, Evanston, Illinois*, 2009.

[109] R. Jiao, J. Bai, X. Liu, T. Sato, X. Yuan, Q. A. Chen, and Q. Zhu, "Learning representation for anomaly detection of vehicle trajectories," *arXiv preprint arXiv:2303.05000*, 2023.

[110] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, Y. Yao, L. Zheng, M. S. Rahman, M. S. Arya, A. Sharma, *et al.*, "The 7th ai city challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5537–5547, 2023.

[111] M. Naphade, Z. Tang, M.-C. Chang, D. C. Anastasiu, A. Sharma, R. Chellappa, S. Wang, P. Chakraborty, T. Huang, J.-N. Hwang, *et al.*, "The 2019 ai city challenge.," in *CVPR workshops*, vol. 8, p. 2, 2019.

[112] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao, "On adversarial robustness of trajectory prediction for autonomous vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15159–15168, 2022.

[113] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023.

[114] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2636–2645, 2020.

[115] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.

[116] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[117] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[118] G. Alinezhad Noghre, V. Katariya, A. Danesh Pazho, C. Neff, and H. Tabkhi, "Pishgu: Universal path prediction network architecture for real-time cyber-physical edge systems," in *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, pp. 88–97, 2023.

[119] N. C. . AFFILIATES, "Software-based power consumption modeling," 2023. Last updated on Aug 11, 2023.