Perception and Prediction in Multi-Agent Urban Traffic Scenarios for Autonomous Driving

by

Prarthana Bhattacharyya

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Florian Shkurti Assistant Professor, Department of Computer Science, University of Toronto
Supervisor(s):	Krzysztof Czarnecki Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Zhou Wang Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Mark Crowley Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Other Member(s):	Steven Waslander Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo Professor, Institute for Aerospace Studies, University of Toronto

Author's Declaration

Statement of Contributions

Prarthana Bhattacharyya is the sole author of this thesis, which was written under the supervision of Prof. Krzysztof Czarnecki. The development of this research resulted in the following publications:

- (i) E Baser, V Balasubramanian*, **P Bhattacharyya***, K Czarnecki. Fantrack: 3d multi-object tracking with feature association network. IEEE Intelligent Vehicles Symposium (IV), 2019.
- (ii) **P Bhattacharyya**, K Czarnecki. Deformable PV-RCNN: Improving 3D Object Detection with Learned Deformations. ECCV 2020 - Perception for Autonomous Driving Workshop.
- (iii) **P Bhattacharyya**, C Huang, K Czarnecki. Sa-det3d: Self-attention based context-aware 3d object detection. ICCV 2021 AVVision Workshop.
- (iv) **P Bhattacharyya**, C Huang, K Czarnecki. SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving. Conference on Robot Learning (CoRL), 2022.
- (v) **P Bhattacharyya**, C Huang, K Czarnecki. SSL-Interactions: Pretext Tasks for Interactive Trajectory Prediction. Preprint, 2023.

Publication (i) is a joint project in which I worked with my colleagues Erkan Baser and Venkateshwaran Balasubramanian. Venkateshwaran Balasubramanian proposed the Siamese architecture with multi-modal inputs. Erkan Baser helped with designing the cosine similarity and cost function and proposed the idea of the probability of occlusion. I proposed the idea of using CNNs for data association. We split the implementation among us. Finally, for publications (iii), (iv) and (v), Chengjie Huang helped revise and refine the manuscripts and prepare the illustrations.

Abstract

In multi-agent urban scenarios, autonomous vehicles navigate an intricate network of interactions with a variety of agents, necessitating advanced perception modeling and trajectory prediction. Research to improve perception modeling and trajectory prediction in autonomous vehicles is fundamental to enhance safety and efficiency in complex driving scenarios. Better data association for 3D multi-object tracking ensures consistent identification and tracking of multiple objects over time, crucial in crowded urban environments to avoid mis-identifications that can lead to unsafe maneuvers or collisions. Effective context modeling for 3D object detection aids in interpreting complex scenes, effectively dealing with challenges like noisy or missing points in sensor data, and occlusions. It enables the system to infer properties of partially observed or obscured objects, enhancing the robustness of the autonomous system in varying conditions. Furthermore, improved trajectory prediction of surrounding vehicles allows an autonomous vehicle to anticipate future actions of other road agents and adapt accordingly, crucial in scenarios like merging lanes, making unprotected turns, or navigating intersections. In essence, these research directions are key to mitigating risks in autonomous driving, and facilitating seamless interaction with other road users.

In Part I, we address the task of improving perception modeling for AV systems. Concretely our contributions are: (i) FANTrack introduces a novel application of Convolutional Neural Networks (CNNs) for real-time 3D Multi-object Tracking (MOT) in autonomous driving, addressing challenges such as varying number of targets, track fragmentation, and noisy detections, thereby enhancing the accuracy of perception capabilities for safe and efficient navigation. (ii) FANTrack proposes to leverage both visual and 3D bounding box data, utilizing Siamese networks and hard-mining, to enhance the similarity functions used in data associations for 3D Multi-object Tracking (MOT). (iii) SA-Det3D introduces a globally-adaptive Full Self-Attention (FSA) module for enhanced feature extraction in 3D object detection, overcoming the limitations of traditional convolution-based techniques by facilitating adaptive context aggregation from entire point-cloud data, thereby bolstering perception modeling in autonomous driving. (iv) SA-Det3D also introduces the Deformable Self-Attention (DSA) module, a scalable adaptation for global context assimilation in large-scale point-cloud datasets, designed to select and focus on most informative regions, thereby improving the quality of feature descriptors and perception modeling in autonomous driving.

In Part II, we focus on the task of improving trajectory prediction of surrounding agents. Concretely, our contributions are: (i) SSL-Lanes introduces a self-supervised learning approach for motion forecasting in autonomous driving that enhances accuracy and generalizability without compromising inference speed or model simplicity, utilizing pseudo-labels from pretext tasks for learning transferable motion patterns. (ii) The second contribution in SSL-Lanes is the design of comprehensive experiments to demonstrate that SSL-Lanes can yield more generalizable and robust trajectory predictions than traditional supervised learning approaches. (iii) SSL-Interactions presents a new framework that utilizes pretext tasks to enhance interaction modeling for trajectory prediction in autonomous driving. (iv) SSL-Interactions advances the prediction of agent trajectories in interaction-centric scenarios by creating a curated dataset that explicitly labels meaningful interactions, thus enabling the effective training of a predictor utilizing pretext tasks and enhancing the modeling of agent-agent interactions in autonomous driving environments.

Acknowledgements

Undertaking this dissertation journey has been among the most demanding yet fulfilling endeavors of my life. I owe immense gratitude several people, encompassing my supervisor, mentors, family, and friends, who made it achievable.

Firstly, my heartfelt gratitude goes to my supervisor, Prof. Krzysztof Czarnecki. His profound insights and unwavering belief in me were pivotal throughout my graduate journey. I am thankful for his guidance and the confidence he placed in me, especially when I ventured into unfamiliar research territory.

I would like to thank the members of my examining committee, Prof. Zhou Wang, Prof. Mark Crowley, Prof. Steven Waslander and Prof. Florian Shkurti for providing very valuable feedback. They have seen me through qualifying examinations and seminars, and have provided thoughtful comments and constructive feedback at various stages of my research. Discussions with them have taught me to think carefully about assumptions and metrics, and reviewing literature critically. In addition, I am thankful for the mentorship of Dr. Michał Antkiewicz, whose guidance and readiness to assist was invaluable to me.

I am grateful to my co-authors, without whom none of this would have been possible. They include, but are not limited to, Dr. Erkan Baser, Venkateshwaran Balasubramanian and Chengjie (Nick) Huang. Nick was instrumental in visually communicating our scientific ideas, designing many of the figures in this dissertation and teaching me design principles that I could apply to future work. I am grateful to Mitacs and Gatik AI for the Mitacs Accelerate Fellowship and the opportunity to collaborate on a topic that would later become a part of my thesis. I would also like to thank Amazon Science and Apple for providing invaluable internship experiences that gave me the opportunity to work with world-class researchers in the field.

I would like to highlight my colleagues in the WISE lab who have made grad-life deeply enjoyable and who I have learned a great deal from - Dr. Jian Deng, Dr. Atrisha Sarkar, Dr. Rodrigo Queiroz, Samin Khan, Matthew Pitropov, Sachin Vernekar, Dr. Vahdat Abdelzad, Edward Chao, Buu Phan, Matthew Angus, Ashish Gaurav, Braden Hurl, Taylor Denouden, Aravind Balakrishnan, Dr. Jaeyoung Lee, Changjian Li, Harry Nguyen, Luke Rowe and Jinwei Zhang. I am also thankful to Dr. Sara Ross-Howe and Dr. Nabiha Asghar, who I had the fortune to meet at the Vector Institute's DL-RL Summer School, and since then participated in many animated discussions about machine learning, research, life, relationships, coffee, food, and what not!

A large part of this thesis was written over the COVID-19 pandemic and a personally challenging time. I consider myself lucky to have had friends across Canada, India, U.K. and Japan, who have always been there for me - Sanjula and Ayush who made me grow roots in Canada; Reza, Yash, Kinjal, Gurshaant (Gary), Xueqing (Caddie), Indrani, Michael, Victoria (Tori), Yuba, Andrea and Karen who made Waterloo less daunting and more entertaining; Soham, Trina, Arunima, who I have come to with many complaints, and the list goes on.

Most importantly I would like to thank my family for their continued understanding and support. This includes Amitvikram Dutta who deserves the most credit for me being able to complete this dissertation and being my best friend and partner; Madhabi Dutta for instilling in me her 'can-do' attitude to life; Dr. Priyanjali Bhattacharyya who drives me to be a better person everyday; Dr. Sarbani Bhattacharyya and Dr. Prasun Bhattacharyya for being my life's biggest inspiration. I can only hope to one day live up to the sacrifices you have made to give me wings and enable me to dream big.

Dedication

To Ma, Baba, Priyanjali and Amitvikram, for their unconditional love, support and kindness.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous driving is an exciting and rapidly evolving field that holds the promise of revolutionizing road transportation. With the potential to significantly enhance road safety, improve traffic efficiency, and increase transportation accessibility, autonomous driving has attracted immense attention from researchers, industry leaders, and policymakers [96]. The continuous advancements in computer vision, deep learning, and sensor technologies and curated large-scale datasets have paved the way for remarkable progress in perception modeling for autonomous vehicles. These vehicles can now perceive their surroundings, detect objects, and make informed decisions based on the analyzed data. However, there are still significant challenges that need to be overcome to achieve the full potential of autonomous driving systems. The deployment of self-driving cars, also known as Level 5 autonomous vehicles [167], may face different theoretical and practical limitations. Instances of accidents involving autonomous vehicles, have also, in rare cases, resulted in fatalities [210, 124].

Better perception modeling in urban multi-object settings is essential to improve the safety and efficiency of autonomous driving. In densely populated urban environments, there is a multitude of dynamic objects, including vehicles, pedestrians, and cyclists. These objects often interact in complex and unpredictable ways. For an autonomous vehicle (AV) to navigate safely, it must accurately detect, identify, and track all such objects in its surroundings. Better *data association* for multi-object tracking is essential because AVs need to maintain an accurate and consistent understanding of each detected object's identity and state (position, velocity, orientation, etc.) over time. Poor data association can lead to identity switches or track terminations, which may cause the AV to lose track of an object's trajectory. For instance, in a scenario where two cyclists cross paths, a failure in data association might lead the AV to lose track of one or both cyclists, potentially resulting in unsafe maneuvers or even collisions. Similarly, *context-aware 3D detection* systems enhance the robustness of autonomous driving systems by effectively handling noisy point clouds, missing points, and occlusions. Missing or noisy points can lead to incomplete or erroneous representation of an object. A car with only its rear half visible might be erroneously classified as a motorcycle. Here, context-aware modeling can infer the full extent of the object by leveraging the typical size and shape of cars and the likely presence of a larger vehicle partially occluded by another object or landscape feature in the scene. Occlusions are particularly challenging in crowded urban scenarios where multiple objects might partially obscure each other. Conventional detection models might struggle to separate overlapping objects. However, context-aware detection systems can infer the presence of fully or partially occluded objects by incorporating the understanding of typical urban scenarios, like a pedestrian obscured by a parked car or a cyclist behind a bus.

In Part I of this dissertation, we focus on two key questions: (a) How can we improve the data-association step for multi-object 3D tracking? (b) How can we improve context modeling in current 3D point-cloud based object-detection systems?

Beyond perception, the capability to reason about the goals and intentions of other road users and accurately predict their future motion is fundamental for AVs. Effective trajectory prediction of surrounding agents is paramount in preventing accidents, particularly in busy urban multi-object environments. Consider an urban multi-lane road where the AV needs to perform a lane change to prepare for a forthcoming turn. Simultaneously, multiple other vehicles around it could also be initiating their own lane changes or overtaking maneuvers. The AV must predict the trajectories of these vehicles, ensuring it has enough space and time to change lanes safely. A miscalculation could lead to potential near-miss situations or accidents. Another common scenario involves the AV needing to make an unprotected left turn across oncoming traffic at a busy intersection. In this case, the AV must predict the speed and trajectory of oncoming vehicles to determine a safe gap for executing the turn. Simultaneously, it should also be aware of vehicles behind that might also be trying to make a similar turn or proceed straight. If the AV incorrectly predicts the trajectories of the oncoming vehicles or misjudges the intent of the vehicle behind, it could result in situations ranging from traffic disruptions to potential collisions.

In Part II of our dissertation, we focus on two key questions: (a) How can we improve the trajectory prediction of surrounding agents, specifically in terms of improving generalization and learning from unlabeled data? (b) How can we improve agent-agent interaction modeling for trajectory prediction?

In the next sections, we discuss our research objectives, themes and contributions in more detail.

1.1 Research Themes

1.1.1 Perception Modeling for Multi-Agent Urban Scenarios

Data-association for 3D Multi-Object Tracking

The central theme of Chapter 3 concerns the development and application of a new methodology for 3D Multi-object Tracking (MOT) in the domain of autonomous driving, particularly focusing on urban environments. The main challenge in such scenarios lies in the data association problem of MOT, where optimal trajectories for multiple objects, or agents, need to be identified and tracked over consecutive frames in real-time. While current strategies, such as 'tracking-by-detection,' have made substantial progress, they still struggle to reach the accuracy of human perception, and face particular difficulty with issues such as varying number of targets, track fragmentation, and noisy detections. This work proposes an innovative solution to these challenges through the implementation of Convolutional Neural Networks (CNNs) in the data association process. By being, to the best of our knowledge, the first work to cast the data association problem as inference within a CNN, the proposed method not only simplifies optimization compared to the conventional Recurrent Neural Networks (RNNs) but also effectively manages the aforementioned issues inherent in real-time, multi-agent urban scenarios. This research, therefore, presents advancement in the perception capabilities for autonomous vehicles, bringing us one step closer to achieving reliable and safe autonomous navigation in complex urban environments.

Similarity Functions for Data-Association

The second research theme of Chapter 3 revolves around enhancing the effectiveness of the data associations in 3D Multi-object Tracking (MOT) by focusing on the improvement of similarity functions used for matching targets and detections. Historically, these functions were built manually utilizing color histograms, bounding box position, and linear motion models, but have shown limitations in their ability to generalize across tasks and complex tracking scenarios. More modern techniques have employed deep neural network architectures for learning pairwise costs, which have yielded better performance across various vision-based tasks. This research advances these techniques by leveraging both visual and 3D bounding box data to develop robust matching costs. The computed pairwise similarities can be used to predict discrete target assignments, as mentioned in the research theme in Sec. 1.1.1. Differing from traditional approaches that use distance functions or hand-crafted features for data association, this study employs Siamese networks to learn generalized and discriminative features from 3D object configurations and visual data. Moreover, the model's objective function is adapted to use cosine-similarity metric with hard-mining, offering a positive impact on the convergence. This simple approach to optimizing the similarity function for data-association in 3D MOT shows promising results for improving the performance of autonomous vehicles' perception capabilities in complex scenarios.

Context-Aware Feature Aggregation for 3D Object Detection

Chapter 4 focuses on the necessity for an improved feature extractor for 3D object detection, a crucial component of perception modeling for autonomous driving. Our research proposes a context-aware module that capitalizes on self-attention mechanisms to overcome the limitations of current convolution-based detection techniques. 3D object detection architectures, including BEV-based, voxel-based, point-based, and point-voxel-based methods, utilize a fixed-weight convolution operator. However, these models, due to the convolution operator's fixed nature, lack the ability to adapt to feature content or focus selectively on salient elements. Additionally, the scaling of parameters with an increased size of the receptive field is inefficient, and capturing global correlations in point-cloud data is challenging. This inspires our exploration of self-attention mechanisms to enhance the context modeling of 3D object detection. While self-attention has been pivotal in achieving superior results in fields like machine translation, image recognition, 2D object detection, activity recognition, person re-identification, and reinforcement learning, its application for global context modeling in point-clouds for 3D object detection remains relatively unexplored. We propose a globally-adaptive Full Self-Attention (FSA) module, which is simple, generic, and permutation-invariant, thereby providing an effective solution to the limitations of convolution-based methods. It facilitates direct communication between remote regions of point-cloud data and enable the learning of relationships across objects. Their application leads to an adaptive aggregation of context information from the entire point-cloud and can be integrated within the backbone of modern point-cloud based detector architectures. The study advances the feature extraction capabilities of 3D object detection architectures, potentially contributing to more robust perception modeling in autonomous driving.

Scalable Context-Aware Feature Aggregation

The second research theme of Chapter 4 focuses on a scalable adaptation of the self-attention module discussed in Sec. 1.1.1. This adaptation is referred to as the Deformable

Self-Attention (DSA) module. The DSA module has been designed to assimilate global context in large-scale point-cloud datasets such as nuScenes and the Waymo Open dataset. The functioning of the DSA module involves selecting a representative subset from the original node vectors to aggregate the global context. The module then up-samples this collected structural information back to all nodes. The subset selection process is made efficient by introducing a geometry-guided vertex refinement procedure inspired by deformable convolution networks. This allows the nodes to adapt and spatially reposition themselves to focus on significant locations that are key for semantic recognition. The DSA module has been trained to gather information primarily from the most informative regions within the point-cloud. This approach substantially enhances the quality of the feature descriptors. Consequently, the DSA module provides improvements in perception modeling for autonomous driving through efficient assimilation of global context in large-scale point-clouds.

1.1.2 Predicting Trajectories of Surrounding Agents

Self-Supervision meets Motion Forecasting

Traditionally, motion prediction has relied on handcrafted rules, kinematic constraints, and road map data, which often fall short in capturing complex long-term behaviors and interactions in diverse scenarios. While recent data-driven methods using vector representations and transformer architectures have shown significant progress, they suffer from complexity and low inference speeds, making them unsuitable for real-world applications. In Chapter 5, we propose to bring self-supervised learning into motion forecasting to improve accuracy and generalizability without compromising inference speed or model simplicity. By leveraging pseudo-labels from pretext tasks, our model learns a more transferable and generalized representation of motion patterns. This does not require additional parameters or computations during inference, thus maintaining an optimal balance of accuracy, simplicity, and efficiency. The approach was demonstrated to be effective on a large scale trajectory prediction benchmark, indicating its potential for improving trajectory prediction in real-world autonomous driving scenarios.

Exploration into why Self-Supervision helps Motion Forecasting

Chapter 5's second contribution involves designing a series of experiments to explore why motion forecasting benefits from self-supervised learning (SSL). We hypothesize that the model acquires richer features from SSL training compared to conventional supervised learning models. We propose that SSL training with pretext tasks improves motion forecasting by assuming feature similarity or smoothness in small map neighborhoods (topology-based context prediction), grouping distant nodes with similar features together (clustering and classification), and by learning richer features from more frequent classes in imbalanced datasets. Our paper's experimental design involves six training and testing setups. The conclusion drawn from these experiments is that there is substantial evidence suggesting that SSL-based tasks yield superior generalization capabilities. Consequently, these tasks demonstrate greater effectiveness than purely supervised training approaches in predicting the trajectories of surrounding agents.

Modeling Agent-Agent Interactions

The research in Chapter 6 introduces a framework that improves interaction modeling for trajectory prediction of surrounding agents. By incorporating pretext tasks designed to model agent interactions, this model reconciles the gap between marginal and joint predictions, striking a balance between computational feasibility and prediction accuracy. The pretext tasks in SSL-Interactions focus on various aspects of agent interaction including range gap, closest distance, direction of movement, and interaction type. This design not only captures crucial interaction-specific patterns but also imparts domain-specific knowledge to the model thereby acting as effective regularizer. A unique advantage of these tasks is their ability to generate pseudo-labels from unlabeled data.

Annotating Interaction Labels

In Sec. 1.1.2, to enable training a predictor that utilizes pretext tasks, we first need to generate a dataset featuring explicit interaction labels. Upon visual examination of a random subset of the data, we uncover several constraints that hinder effective interaction modeling. For instance, in certain cases, the target agent is the only vehicle in a large surrounding area, which severely limits potential interactions. In other scenarios, the target agent remains unaffected by nearby vehicles due to factors like distance or lack of spatio-temporal conflict. It's evident that a model trained solely on this data would offer little benefit in terms of interaction modeling. Consequently, to enhance our model's capacity for capturing interactions, we compile a dataset specifically crafted to address these issues and incorporate instances of meaningful interactions in Chapter 6. This addresses the importance of annotating interaction-specific data for predicting surrounding agent trajectories in interaction-centric scenarios.

1.2 Contributions and Outline

In summary, the autonomous driving system involves several interconnected components: perception, prediction, planning, and control. This dissertation focuses on the first two components, perception and prediction, specifically as they pertain to 3D object detection, multi-object tracking, and motion forecasting. We begin this dissertation with an exploration of foundational literature in Chapter 2. We delve into relevant research that serves as the basis for our proposed work. In this chapter, we also draw clear distinctions between our approach and the existing literature, underscoring the unique aspects of our proposed methodology. We then present our 3D multi-object tracker FANTrack in Chapter 3 followed by our improved 3D object detector SA-Det3D in Chapter 4. Chapter 5 presents SSL-Lanes, which is the first principled study to combine self-supervised learning and motion forecasting. In Chapter 6 we propose pretext tasks to improved interaction modeling for motion forecasting. Finally, Chapter 7 ends this dissertation with a conclusion and future directions that are yet to be explored for perception and prediction in autonomous driving. The key contributions of each chapter are summarized as follows.

1.2.1 FANTrack: 3D Multi-Object Tracking with Feature Association Network

In Chapter 3, we propose a data driven approach to online multi-object tracking (MOT) that uses a convolutional neural network (CNN) for data association in a tracking-by-detection framework. The problem of multi-target tracking aims to assign noisy detections to a-priori unknown and time varying number of tracked objects across a sequence of frames. A majority of the existing solutions focus on either tediously designing cost functions or formulating the task of data association as a complex optimization problem that can be solved effectively. Instead, we exploit the power of deep learning to formulate the data association problem as inference in a CNN. To this end, we propose to learn a similarity function that combines cues from both image and spatial features of objects. Our solution learns to perform global assignments purely from data, handles noisy detections and varying number of targets, and is easy to train. We evaluate our approach on the challenging KITTI [43] dataset and show competitive results. FANTrack is associated with the following conference publication [9]:

E Baser, V Balasubramanian*, **P Bhattacharyya***, K Czarnecki. Fantrack: 3d multi-object tracking with feature association network. IEEE Intelligent Vehicles Symposium (IV), 2019.

1.2.2 SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection

Existing point-cloud based 3D object detectors use convolution-like operators to process information in a local neighbourhood with fixed-weight kernels and aggregate global context hierarchically. However, non-local neural networks and self-attention for 2D vision have shown that explicitly modeling long-range interactions can lead to more robust and competitive models. In Chapter 4, we propose two variants of self-attention for contextual modeling in 3D object detection by augmenting convolutional features with self-attention features. We first incorporate the pairwise self-attention mechanism into the current state-of-the-art BEV, voxel and point-based detectors and show consistent improvement over strong baseline models of up to 1.5 3D AP while simultaneously reducing their parameter footprint and computational cost by 15-80% and 30-50%, respectively, on the KITTI validation set. We next propose a self-attention variant that samples a subset of the most representative features by learning deformations over randomly sampled locations. This not only allows us to scale explicit global contextual modeling to larger point-clouds, but also leads to more discriminative and informative feature descriptors. Our method can be flexibly applied to most state-of-the-art detectors with increased accuracy and parameter and compute efficiency. We show our proposed method improves 3D object detection performance on KITTI [43], nuScenes [19] and Waymo Open datasets [156]. SA-Det3D is associated with the following publications [14, 15]:

P Bhattacharyya, K Czarnecki. Deformable PV-RCNN: Improving 3D Object Detection with Learned Deformations. ECCV 2020 - Perception for Autonomous Driving Workshop.

P Bhattacharyya, C Huang, K Czarnecki. Sa-det3d: Self-attention based context-aware 3d object detection. ICCV 2021 AVVision Workshop.

1.2.3 SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving

Self-supervised learning (SSL) is an emerging technique that has been successfully employed to train convolutional neural networks (CNNs) and graph neural networks (GNNs) for more transferable, generalizable, and robust representation learning. However its potential in motion forecasting for autonomous driving has rarely been explored. In Chapter 5, we report the first systematic exploration and assessment of incorporating self-supervision into motion forecasting. We first propose to investigate four novel self-supervised learning tasks for motion forecasting with theoretical rationale and quantitative and qualitative comparisons on the challenging large-scale Argoverse [26] dataset. Secondly, we point out that our auxiliary SSL-based learning setup not only outperforms forecasting methods which use transformers, complicated fusion mechanisms and sophisticated online dense goal candidate optimization algorithms in terms of performance accuracy, but also has low inference time and architectural complexity. Lastly, we conduct several experiments to understand why SSL improves motion forecasting. SSL-Lanes is associated with the following conference publication [16]:

P Bhattacharyya, C Huang, K Czarnecki. SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving. Conference on Robot Learning (CoRL), 2022.

1.2.4 SSL-Interactions: Pretext Tasks for Interactive Trajectory Prediction

In Chapter 6, we address motion forecasting in multi-agent environments, crucial for safety of autonomous vehicles. Traditional prediction methods struggle to accurately model non-linear interactions while recent popularly-used data-driven marginal trajectory prediction methods struggle to properly learn agent-to-agent interactions. In this chapter, we present SSL-Interactions, a framework that proposes pretext tasks to enhance interaction modeling in trajectory prediction. These tasks encapsulate various aspects of interaction from domain-specific knowledge which can support better model regularization, and also provides a computationally efficient solution to balance marginal and fully-joint predictions. We introduce four interaction-aware pretext tasks—range gap prediction, closest distance prediction, direction of movement prediction, and type of interaction prediction. We further propose an approach to curate interaction-specific scenarios from datasets and explicitly label pairs of interacting agents. This facilitates generation of pseudo-labels for interaction-centric pretext tasks. The use of pretext tasks and pseudo-labels allows the model to learn from unlabeled data. We also propose three new metrics specifically designed to evaluate predictions in interactive scenes. Our empirical evaluations indicate SSL-Interactions outperforms state-of-the-art motion forecasting methods both quantitatively (with up to 8% improvement) and qualitatively for interaction-heavy scenarios on the Argoverse [26] dataset. SSL-Interactions is associated with the following preprint:

P Bhattacharyya, C Huang, K Czarnecki. SSL-Interactions: Pretext Tasks for Interactive Trajectory Prediction. Preprint, 2023.

Chapter 2

Related Work

This chapter provides an overview of the related work that this dissertation is built upon. This research is primarily related to three major topics: 3D Multi-Object Tracking, discussed in Sec. 2.1.1; 3D Object Detection, discussed in Sec. 2.1.5; and Trajectory Prediction, as discussed in Sec. 2.2. We emphasize the unique aspects of our proposed work, contrasting it with existing studies within each section.

2.1 Perception Modeling for Multi-Agent Urban Scenarios

2.1.1 3D Multi-Object Tracking

In this section, we provide a concise review of recent advancements in the domain of data association for multi-object tracking. Additionally, we discuss the methods employed for quantifying similarity within the context of data association. We finally discuss how our propose multi-object tracking approach advances the literature.

2.1.2 Data Association in MOT

The data association problem forms the core of the multi-object tracking problem. At time instant τ, let us consider a set of objects that have been observed n the past, that is, from the first frame until τ. The set of trajectories corresponding to these objects are called targets or tracks (T). The set of objects observed at time τ are called measurements (M).Data association finds assignments of the form (t_i, m_j), where $t_i \in T$ and $m_j \in M$ are the i_{th} target and j_{th} measurement, respectively, and not two tracks are assigned the same measurement. This is done by minimizing a cost function or maximizing the similarity metric between the corresponding target and the measurement. Data association gives us pairs that have the highest similarity scores, which helps in the encompassing problem of tracking.

Classical approaches solve the data association problem by considering multiple hypotheses for an assignment (MHT) [5], or by jointly considering all possible assignment hypotheses (JPDA) [37]. These formulations prove to be very computationally intensive, however. Many recent works process sequences in *batch* mode, using a graph-based representation with detections as nodes and possible assignments as edges. The optimization is then cast as a linear program solved to (near) global optimality with relaxation, min-cost or shortest path algorithms [18, 201, 176]. More complex optimization schemes include MCMC [72] and discrete-continuous settings [4]. However, *global* optimization formulations are unsuited to real-time applications like autonomous navigation.

Online methods estimate the current state using the information only from the past frames and the current one. Commonly used state-estimators include the Kalman filter [71] for linear motion and particle filters [49] for multi-modal posteriors. The two-frame association problem is often solved using a greedy or Hungarian algorithm [80]. Approaches based on local associations tend to be susceptible to track fragmentation and noisy detections, however.

Deep learning has achieved state-of-the-art results in perception tasks like image classification, segmentation, and single object tracking. Milan et al. proposed the first fully end-to-end multi-object tracking method based on deep learning. The method predicts the assignment of each target, one at a time, using an RNN [106].

2.1.3 Measuring Similarity

Tracking algorithms have used distance functions such as Euclidean [148] and Mahalanobis distance [39] as matching costs for data association. Other similarity measures include color-based appearance features [134], SIFT-like features [101], and linear and non-linear motion models and their various weighted combinations [31]. These tediously hand-crafted features fail to generalize across complex scenarios and backgrounds, however. Recent works explore learning pairwise costs using deep structured SVM [1], CNNs [151], and RNNs [136]. For CNNs, similarity learning often exploits Siamese networks. Leal-Taixe̋ et al. [84] and Frossard et al. [40] use them to learn descriptors for matching with multi-modal inputs.

2.1.4 FANTrack relative to literature

We propose FANTrack in Chapter 3. In contrast to the current multi-object tracking systems [40, 106, 84], our approach feeds all detections and their learned similarity scores at once into a CNN to predict the assignments. Our model is easier to optimize than an RNN, handles noisy detections and a varying number of targets, and considers all targets at once when performing assignments. While we also use Siamese networks to learn *generalized* and *discriminative* features from 3D object configurations and visual information conditioned on similarity, we adapt our objective function to use the cosine-similarity metric with hard-mining which has a positive impact on convergence.

2.1.5 3D Object Detection

Current 3D object detectors include BEV, voxel, point or hybrid (point-voxel) methods. *BEV-based* methods like MV3D [29] fuse multi-view representations of the point-cloud and use 2D convolutions for 3D proposal generation. PointPillars [83] proposes a more efficient BEV representation and outperforms most fusion-based approaches while being 2-4 times faster. *Voxel-based* approaches, on the other hand, divide the point-cloud into 3D voxels and process them using 3D CNNs [207]. SECOND [184] introduces sparse 3D convolutions for efficient 3D processing of voxels, and CBGS [208] extends it with multiple heads. *Point-based* methods are inspired by the success of PointNet [129] and PointNet++ [130]. F-PointNet [128] first applied PointNet for 3D detection, extracting point-features from point-cloud crops that correspond to 2D camera-image detections. Point-RCNN [144] segments 3D point-clouds using PointNet++, and uses the segmentation features to better refine box proposals. *Point-Voxel-based* methods like STD [191], PV-RCNN [143] and SA-SSD [55] leverage both voxel and point-based abstractions to produce more accurate bounding boxes.

2.1.6 Attention for Context Modeling

Self-attention [164] has been instrumental to achieving state-of-the-art results in machine translation and combining self-attention with convolutions is a theme shared by recent work in natural language processing [174], image recognition [10], 2D object detection

Method	Task	Modality	Context	Scalability	Attention + Convolution Combination	Stage Added
HG-Net [27]	detection	points	global-static	-	gating	Attention modules are added at the end.
PCAN [203]	place-recognition	points	local-adaptive	-	gating	
Point-GNN [146]	detection	points	local-adaptive	-	-	Attention modules fully replace convolution and set-abstraction layers.
GAC [168]	segmentation	points	local-adaptive	-	-	
PAT [187]	classification	points	global-adaptive	randomly sample points subset	-	
ASCN [182]	segmentation	points	global-adaptive	randomly sample points subset	-	
Pointformer [120]	detection	points	global-adaptive	sample points subset and refine	-	
MLCVNet [181]	detection	points	global-static	-	residual addition	Attention modules are inserted into the backbone.
TANet [100]	detection	voxels	local-adaptive	-	gating	
PMPNet [192]	detection	pillars	local-adaptive	-	gated-recurrent-unit	
SCANet [102]	detection	BEV	global-static	-	gating	
A-PointNet [117]	detection	points	global-adaptive	attend sequentially to small regions	gating	
Ours (FSA/DSA)	detection	points, voxels, pillars, hybrid	global-adaptive	attend to salient regions using learned deformations	residual addition	Attention modules are inserted into the backbone.

Table 2.1: Properties of recent attention-based models for point-clouds

[114], activity recognition [169], person re-identification [204] and reinforcement learning [195].

Using self-attention to aggregate global structure in point-clouds for 3D object detection remains a relatively unexplored domain. PCAN [203], TANet [100], Point-GNN [146], GAC [168], PMPNet [192] use local context to learn context-aware discriminative features. However relevant contextual information can occur anywhere in the point-cloud and hence we need global context modeling. HGNet [27], SCANet [102], MLCVNet [181] use global scene semantics to improve performance of object detection, but the global context vector is shared across all locations and channels and does not adapt itself according to the input features leading to a sub-optimal representation. PAT [187], ASCN [182], Pointformer [120] build globally-adaptive point representations for classification, segmentation and 3D detection. But because they use the costly pairwise self-attention mechanism, the self-attention does not scale to the entire point-cloud. Consequently, they process a randomly selected subset of points, which may be sensitive to outliers. To process global context for 3D object detection and scale to large point-clouds, Attentional PointNet [117] uses GRUs [32] to sequentially attend to different parts of the point-cloud. Learning global context by optimizing the hidden state of a GRU is slow and inefficient, however.

2.1.7 SA-Det3D relative to literature

Instead of repeatedly stacking convolutions, we propose SA-Det3D in Chapter 4. It consists of a simple, scalable, generic and permutation-invariant block called FSA/DSA to adaptively aggregate context information from the entire point-cloud. This allows remote regions to directly communicate and can help in learning relationships across objects. This module is flexible and can be applied in parallel to convolutions within the backbone of modern point-cloud based detector architectures. Our method can also processes context *adaptively* for each location from the entire point-cloud, while also *scaling* to large sets using learned deformations. Since the global context is fused with local-convolutional features, the training is stable and efficient as compared to GRUs or stand-alone attention networks [132]. Tab. 2.1 compares our work with recent point-cloud based attention methods.

2.2 Predicting Trajectories of Surrounding Agents

In this section, we provide an overview of recent developments in the field of multi-agent trajectory prediction. We also briefly describe literature related to self-supervised learning. Furthermore, we discuss the application of pretext tasks for the purpose of interaction modeling. Lastly, we elaborate on how our proposed motion forecasting systems augment the current state-of-the-art methods.

2.2.1 Motion Forecasting

Traditional methods for motion forecasting primarily use Kalman filtering [71] with a prior from HD-maps to predict future motion states [180, 60]. With the huge success of deep learning, recent works use data-driven approaches for motion forecasting. These methods explore different architectures involving rasterized images and CNNs [25, 8, 125], vectorized representations and graph neural networks (GNNs) [41, 197, 107, 105, 73], point-cloud representations [189], transformers [111, 98, 47, 66] and sophisticated fusion mechanisms [88], to generate features that predict final output trajectories. While the focus of these works is to find more effective ways of feature extraction from HD-maps and interacting agents, they need huge model capacity, heavy parameterization, and extensive augmentations or large amounts of data to converge to a general solution. Other works [205, 24, 152, 198] build on them to incorporate prior knowledge in the form of predefined candidate trajectories from sampling or clustering strategies from training data. However the disadvantage of these methods is that their performance is highly related to the quality of the trajectory proposals, which becomes an extra dependency. End-to-end solutions for optimizing end-points of these candidates trajectories are proposed by Dense-TNT [51] and HOME [45]. Dense-TNT has state-of-the-art accuracy with a reasonable parameter budget, but its online dense goal candidate optimization strategy is computationally very expensive, which is unrealistic for real-time operations like autonomous driving. Lately, ensembling techniques like MultiPath++ [163] and DCMS [190] have been proposed. While they have high forecasting performance, a major disadvantage is their high memory cost for training and heavy computational cost at inference.

2.2.2 Self-supervised Learning

SSL is a rapidly emerging learning framework that generates additional supervised signals to train deep learning models through carefully designed pretext tasks. In the image domain, various self-supervised learning techniques have been developed for learning high-level image representations, including predicting the relative locations of image patches [35], jigsaw puzzle [112], image rotation [44], image clustering [20], image inpainting [123], image colorization [202] and segmentation prediction [122]. In the domain of graphs and graph neural networks, pretext tasks include graph partitioning, node clustering, context prediction and graph completion [193, 70, 99, 64].

2.2.3 Interactive Trajectory Prediction

In recent years, approaches driven by data have shown superior results as they learn interactions directly from the input, particularly in the context of real-world driving scenarios. To predict interactive pedestrian trajectories in crowded scenes, Social-LSTM[3] and Social-GAN[53] leverage social pooling mechanisms that effectively capture social influences from neighboring agents. These mechanisms enable the model to incorporate the collective behavior of nearby individuals when making trajectory predictions. Graph neural networks (GNNs) possess robust relational inductive biases and have exhibited remarkable performance in tasks that involve relational reasoning, including visual question answering and complex physical systems [138]. GNNs have been extensively used in traffic scenarios to model interactions between agents [88, 22, 23, 137]. [111, 159] typically use a form of attention mechanism, which is a special case of GNNs, that allow the model to weigh the importance of different entities in the scene relative to each other. GNNs implicitly model agent-to-agent interactions by structuring the data as a graph, where each node represents an agent and edges signify the relationship or interaction between agents. Through propagation and update rules, GNNs can capture the influence of one agent on another during prediction, encoding the interactive dynamics *implicitly* within the traffic scenario. However, this prediction is marginal because it's made independently for each agent, considering only the current state of its neighbors in the graph. It does not account for the joint interaction effects that could arise from the simultaneous movement of multiple agents.

Joint prediction of trajectories in a traffic scene, while providing a comprehensive view of the interactions between multiple agents, comes with its own set of challenges. It requires modeling the dependencies between all agents, which can increase the computational complexity exponentially with the number of agents, making it impractical for real-time applications. [154] proposes utilizing *explicit* latent interaction graphs via multiplex attention to infer high-level abstractions for improved multi-agent system forecasting. The M2I model [157] leverages a marginal predictor to produce predictive samples for the 'influencer' agents, and employs a conditional predictor to project the future trajectories of the 'reactor' agents based on the influencer's anticipated path. This approach, however, requires the training of a relation predictor capable of discerning the influencer and reactor roles even during the inference stage. This may introduce noise and potentially struggle to scale effectively when dealing with multiple interactive agents.

2.2.4 Pretext tasks for interactive modeling

In related research, the use of pretext tasks has enhanced the modeling of interactions and relational reasoning across different contexts. [82] demonstrates that predicting language descriptions and explanations as an auxiliary task can significantly enhance reinforcement-learning (RL) agents' abilities to infer abstract relational and causal structures in complex environments. [81] introduces a trajectory prediction model that utilizes linguistic intermediate representations to enhance forecasting accuracy and model interpretability. [155] uses pseudo-labels in a multi-agent trajectory prediction setting to induce an informative, interactive latent space for a conditional variational auto-encoder (CVAE), thereby mitigating posterior collapse and improving the trajectory prediction accuracy. The work most similar to ours is Social-SSL [162], which leverages self-supervised pre-training to enhance data efficiency and the generalizability of Transformer networks in pedestrian only trajectory prediction. Nonetheless, its proposed pretext tasks do not necessarily exert a direct influence on the subsequent motion forecasting task. This is primarily due to the lack of a strong correlation between success in their pretext tasks and enhanced performance in motion forecasting. For example, predicting potentially non-informative regions of a trajectory through auxiliary loss may not consistently result in downstream performance improvement. Moreover, Social-SSL calculates pretext losses across all agents within a given scene, regardless of whether they are actively interacting. For example, a vehicle maintaining a straight course may not be significantly influenced by an oncoming vehicle if they occupy distinct lanes.

2.2.5 SSL-Lanes relative to literature

To the best of our knowledge, SSL-Lanes proposed in Chapter 5 is the first principled approach that explores motion forecasting for autonomous driving with self-supervision. We use this sub-section to distinguish our work from methods that we believe have similar intuition but very different construction, in order to highlight its novelty and value.

- SSL-Lanes vs. VectorNet [41]: Vector-Net is the only other motion forecasting work that proposes to randomly mask out the input node features belonging to either scene context or agent trajectories, and ask the model to reconstruct the masked features. Their intuition is to encourage the graph networks to better capture the interactions between agent dynamics and scene context. However, our motivation differs from VectorNet in two respects: (a) We propose to use masking to learn local map-structure better, as opposed to learning interactions between map and the agent. This is an easier optimization task, and we out-perform VectorNet. (b) A lane is made up of several nodes. We propose to randomly mask out a certain percentage of each lane. This is a much stronger prior as compared to randomly masking out any node (which may correspond to either a moving agent or map) and ensures that the model pays attention to all parts of the map.

- SSL-Lanes vs. CS-LSTM [34]: CS-LSTM appends the encoder context vector with a one-hot vector corresponding to the lateral maneuver class and a one-hot vector corresponding to the longitudinal maneuver class. Subsequently, the added maneuver context allows the decoder LSTM to generate maneuver specific probability distributions. This construction however is quite different from our work because it is not auxiliary in nature - it always outputs and appends a maneuver to the decoder, even during inference. This we believe is too strong of a bias for the prediction model, especially given the fact that the maneuvers are generated using very simple velocity profiles and not from careful mining of the data. In our conditioning, the maneuvers are mined from data and the final motion prediction does not depend directly on them. We believe this design is much more flexible since it allows to generate more supervisory signals in the form of maneuvers during training, but at the same time does not require an explicit maneuver to condition the final future forecast trajectory output during inference.

- SSL-Lanes vs. MultiPath [25]: MultiPath is also not auxiliary in nature: it factorizes motion uncertainty into intent uncertainty and control uncertainty; models the uncertainty over a discrete set of intents with a softmax distribution; and then outputs control uncertainty as a Gaussian distribution dependent on each waypoint state of the anchor trajectory (corresponding to the intent). While this construction is highly intuitive and effective by design, it is very different from our SSL-based construction. Ours is an auxiliary task which provides supervision during training, and effectively functions as a regularizer, while being general enough to be used with any other data-driven motion forecasting model.

2.2.6 SSL-Interactions relative to literature

SSL-Interactions proposed in Chapter 6 is a framework that factorizes and scales joint distribution prediction into marginal prediction and pretext task distribution prediction during the training stage. This is in contrast to M2I [157] which leverages a marginal predictor to produce predictive samples for the 'influencer' agents, and employs a conditional predictor to project the future trajectories of the 'reactor' agents based on the influencer's anticipated path. This approach, however, requires the training of a relation predictor capable of discerning the influencer and reactor roles even during the inference stage. This may introduce noise and potentially struggle to scale effectively when dealing with multiple interactive agents.

Marginal predictions, which are usually only implicitly considering interaction information for predicting future trajectories, may not be sufficiently influential in an end-to-end learning setting aimed at optimizing future prediction [88, 3]. SSL Interactions is conditioned to consider this interaction information explicitly during motion forecasting, thereby improving interaction modeling.

The proposed pretext tasks for SSL-Interactions, driven by domain-specific knowledge, are closely linked to the efficacy of downstream motion forecasting, in contrast to proxy tasks proposed in [81, 155, 162]. Our pretext tasks not only enforce interaction regularization in the data-driven forecasting model but also come with the advantage of not adding any additional parameters during the inference phase. Significantly, they also offer the capability to learn from unlabeled data.

Part I

Perception Modeling for Multi-Agent Urban Scenarios

Chapter 3

FANTrack: 3D Multi-Object Tracking with Feature Association Network

3D object tracking is of great importance for autonomous driving systems because it not only identifies objects within the environment but also monitors their movement over time. By continuously tracking detected objects, it provides insights into the dynamic nature of the surroundings. This capability allows autonomous systems to make more accurate predictions about the future states of these objects, thus facilitating safer and more efficient decision-making. In this chapter, we describe our proposed 3D object tracker FANTrack [9]. This tracker is specifically designed to enhance the data association step in multi-object tracking, thereby resulting in superior tracking performance. FANTrack code and documentation is open-source.[1]

3.1 Motivation and Contributions

Multi-object tracking (MOT) is the problem of finding the optimal set of trajectories of objects of interest over a sequence of consecutive frames. Most of the successful computer vision approaches to MOT have focused on the *tracking-by-detection* principle [115, 172]. This paradigm allows the problem to be divided into two steps. First, an object detector is used to identify the potential locations of objects in the form of bounding boxes, and

[1]https://git.uwaterloo.ca/wise-lab/fantrack

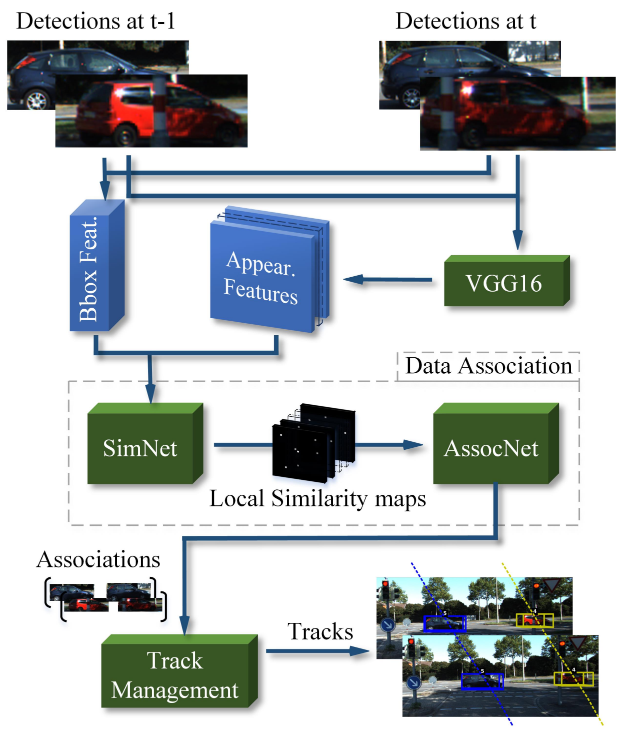


Figure 3.1: Overall architecture of the proposed FANTrack

then a discrete combinatorial problem is solved to link these noisy detections over time to form trajectories. Despite decades of research, the status quo of tracking is far from reaching human accuracy. Current challenges to the problem include a varying and a-priori unknown number of targets; incorrect and missing detections; changing appearances of targets due to sensor motion, illumination, and angle of view; frequent occlusions, and abrupt changes in motion.

The linking step called *data association* is arguably the most difficult component of MOT. Traditional *batch* methods usually formulate MOT as a *global* optimization problem, with the assumption that detections from all future frames are available, and solve it by mapping it to a graph based min-cost flow algorithm [1, 12]. *Online* Markovian formulations of MOT on the other hand often employ greedy or bipartite graph matching methods like the Hungarian algorithm to solve the assignment problem [109, 158, 17]. *Online* approaches are well suited to real-time applications such as tracking road-traffic participants. The success of the final associations is also dependent on the similarity functions used to match the targets and detections. Traditionally cost functions have been handcrafted with representations based on color histograms, bounding box position, and linear motion models [74, 113], but have failed to generalize across tasks and for complex tracking scenarios. Recently, deep neural network architectures have shown superior performance in many vision based tasks. Milan et al. proposed the first end-to-end formulation for MOT, using a recurrent neural network (RNN) to solve the assignment problem for each target independently based on Euclidean cost [106]. However, the use of convolutional neural networks (CNNs), which are easier to train than RNNs, in order to solve the association problem while also learning the cost function has not yet been investigated.

In this chapter, we propose an *online* MOT formulation that casts the assignment problem as inference in a CNN. We present a two-step learning based approach (see Fig. 3.1). The first step learns a similarity function that takes advantage of both visual and 3D bounding box data to yield robust matching costs. The second step trains a CNN to predict discrete target assignments from the computed pair-wise similarities. The benefit of our proposal is that it is easy to train, takes care of a varying number of targets and noisy detections, and provides a simple way to consider all the targets while making associations.

Contributions: We empirically demonstrate on the KITTI tracking dataset [43] that:

- Our approach can solve the multi-target association problem by performing inference using CNNs.
- It can integrate image based appearance and 3D bounding box features to get a discriminative as well as generalized feature representation, thereby learning a robust cost function for association.
- We show competitive qualitative and quantitative 3D tracking results compared to the state of the art.

3.2 FANTrack

Our proposed framework is based on tracking by detection paradigm. Our problem setup assumes at any time instant t we have N number of targets, M number of detections and track labels for every i^{th} track. We use AVOD [79] as our 3D object detector since it achieves state-of-the-art results on KITTI and is open-source, but in principle, any other 3D object detector could be used. The motivation for building FANTrack is to leverage the power of Siamese networks to model the similarities between targets and detections, CNNs to solve the data association problem in MOT, and an online track management module to update, initialize and prune tracks. We describe these modules in the following sections.

3.2.1 SimNet: Similarity Network

The similarity network, also called SimNet, is the first network in our combined architecture for data association, as shown in Fig. 3.2. The main role of the network is to compute a similarity metric between every target and measurement pair and to structure the results as localized maps, which are then used by another network for data association. The SimNet uses three instances of a Siamese Network architecture, one for each of the two types of features (i.e., bounding boxes and appearance), and a third one for weighting the relative importance of each feature. The input to the SimNet consists of bounding box parameters and appearance features of the targets and measurements. We follow the approach of combining each of the two Siamese branches in the cost function [84] [36], by separately processing the target and measurement specific inputs. Though this approach leads to slightly reduced accuracy [84], it is faster than the approach of input stacking. The *output* from *SimNet* is a set of maps called local similarity maps, with dimensions 21x21 pixels and N_{max} channels, one for each target from the $t-1$ frame. Each map is centered around the corresponding target and records the similarity score of the target with every measurement within the $10\,\text{m} \times 10\,\text{m}$ bird's eye view region around the target. The resolution of the maps is $0.5\,\text{m}$.

The SimNet has two main branches: a *bounding box branch* and an *appearance branch*. Each of the two branches, being a Siamese network, applies the same set of weights to each of its input, i.e., target or measurement representation, separately and outputs the corresponding normalized feature vector for each of these two inputs. The respective contribution of the bounding box and appearance branch towards the final similarity score computation is weighted using the *importance branch*. Finally, the weighted average of the cosine-similarities of each target-measurement unit vector pairs for each branch are computed and the scalars are mapped to their corresponding positions on the above-mentioned set of local maps. We describe the individual branches and the similarity and map generation processes in detail in the subsequent subsections.

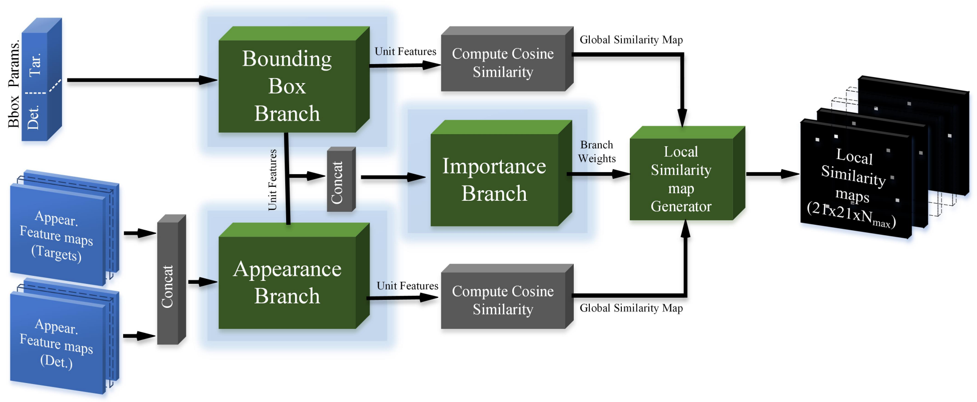


Figure 3.2: Architecture of the proposed Siamese network for similarity learning. The branches highlighted in blue have trainable parameters.

A. Bounding Box Branch

The bounding box branch outputs normalized vectors, each representing a target or measurement bounding box. The cosine similarity between any one of the target unit vectors and any one of the measurement unit vectors is obtained as the dot product of the two vectors. We train a Siamese network with input pairs of target and detection 3D bounding boxes for this purpose. The 3D bounding boxes are defined by their centroids (x, y, z), dimensions (l, w, h), and rotation around the z-axis (θ_z) in the ego-car's IMU/GPS coordinates. To prevent learning variations induced due to ego-motion, detection centroids are converted to coordinates at a common time-step using GPS data. The bounding box parameters are obtained from the object detector (AVOD) and are represented as follows,

$$[x, y, z, l, w, h, ry] \tag{3.1}$$

The input to this branch is an $(N+M)\times 1\times 7$ tensor for N targets and M measurements where the third dimension consists of the 7 bounding box parameters defined above. The inputs are fed to a convolutional layer with 256 1×1 filters to capture complex interactions across the 7 channels, followed by two convolutional layers with 512 1×1 filters. Thus, the parameters of each target and detection are processed independently in order to be later combined in the loss function [93]. We apply L2 normalization on the output features and henceforth refer to the result as *unit* features. The unit features have dimensions $(N + M) \times 512$. These unit features are sliced on the first dimension according to the number of targets T to obtain target and measurement bounding box features. These sliced

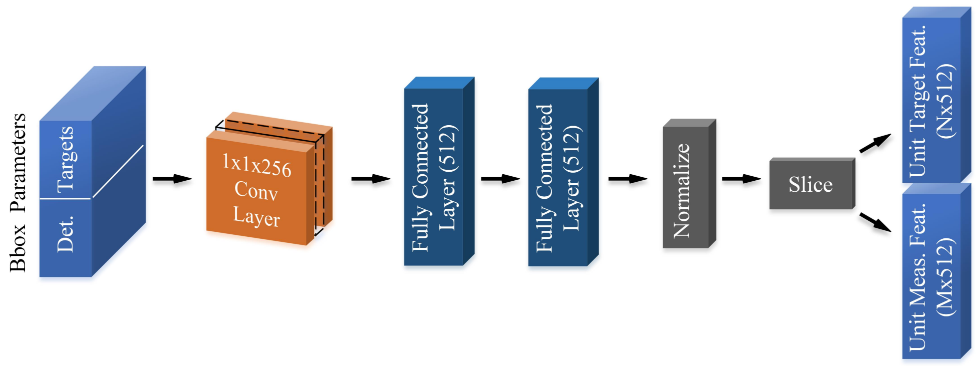


Figure 3.3: Detailed architecture of the bounding box branch. The inputs are bounding box parameters of N targets and M measurements. In training, N and M are 128 (batch size) respectively. Outputs are sliced unit feature vectors.

unit features along with those obtained from the appearance branch are used to compute the cosine similarities, which will be discussed in the subsequent sections. We use batch normalization and leaky-ReLU activation across all the layers. A detailed architecture of the bounding box branch is shown in Fig. 3.3.

B. Appearance Branch

The appearance branch is used to analyze the 2D visual cues in the targets and measurements for the computation of the similarity scores. We train another Siamese network for this purpose. The input to this branch are the appearance features produced by a VGG16 image feature extractor pre-trained on the ImageNet dataset. The dimensions of the input appearance feature maps are (7x7x640). The architecture of the branch is shown in Fig. 3.4. First, we apply 256 3×3 convolutions to obtain promising features for similarity learning while preserving the spatial size of the input. Before flattening the feature maps for the fully-connected layers with 512 neurons, the Global Average Pooling (GAP) [93] layer extracts one abstract feature from each feature map. Similar to the bounding box branch, L2 normalization yields a vector of dimension $(N+M) \times 512$. As in the case of the bounding box branch, the $(N+M) \times 512$ features are sliced along the *first* dimension to obtain the unit appearance features of detections and targets. These are used to compute the appearance cosine similarities.

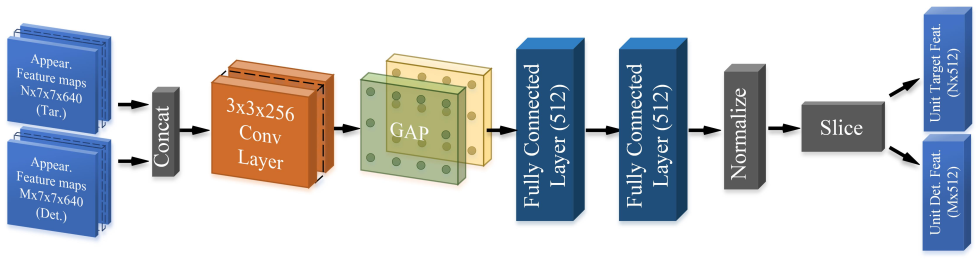

Appear. Feature maps Nx7x7x640 (Tar.)
Appear. Feature maps Mx7x7x640 (Det.)
Concat
3x3x256 Conv Layer
GAP
Fully Connected Layer (512)
Fully Connected Layer (512)
Normalize
Slice
Unit Target Feat. (Nx512)
Unit Det. Feat. (Mx512)

Figure 3.4: Detailed architecture of the appearance branch

C. Importance Branch

The aim of the importance branch is to determine the *relative importance* of the bounding box and appearance features in the computation of the final cosine similarity score. The inputs to this branch are the unit bounding box and unit appearance features of both targets and measurements as shown in Fig. 3.5. First, the vector representation of objects obtained from the appearance and bounding box branches are concatenated to form a single vector (dimension 1024). This, in turn, is connected to a fully-connected layer having two neurons. Finally, the softmax layer computes two scalars representing the importance weights as probabilities of each of the two branches. The weights are sliced along the first and second dimensions to obtain the appearance and bounding box branch weight pair individually for each target t and measurement m. For a given target and measurement pair, the appearance/bounding box weight is aggregated as follows

$$\omega_\beta = \frac{\omega_\beta^{(t)} \times \omega_\beta^{(m)}}{\Omega} \, for \; \beta \in \{bbox, appear\}, \tag{3.2}$$

where Ω is the normalization constant which ensures that the weights are summed up to unity.

D. Similarity Maps

The output of the SimNet are N_{max} local similarity maps, one map per target. The choice of the parameter N_{max} is a design choice, and it directly relates to the maximum number of objects the tracker can associate at a given time instant. Initially, N occupancy maps of measurement indices are constructed on the Bird's Eye View IMU/GPS frame. The resolution of the maps is chosen as 0.5 m, and with a range of 10 m × 10 m, centered around

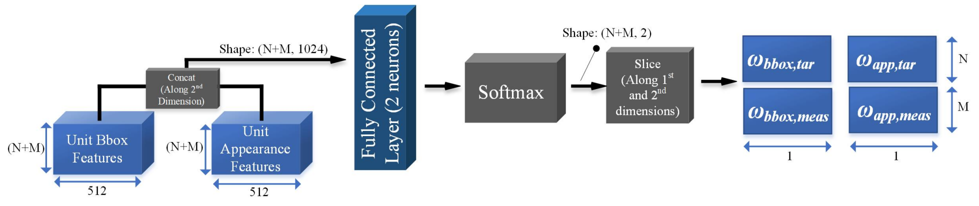


Figure 3.5: Detailed architecture of the importance branch

the respective targets locations we get maps of size 21×21 pixels. Next the unit feature vectors of size 512 computed by the similarity network for each measurement in the maps are placed in the respective measurement locations. For every local map the corresponding target's unit target feature vector of size 512 is convolved with the measurement features on the locations where the measurement is present. In effect, this operation computes the dot product of the unit vectors, which is their cosine similarity. This selective convolution operation for a typical map channel is shown in Fig. 3.6.

The process of building these maps is done for bounding box and appearance branches, yielding two sets of maps. Next, two weight maps are built by placing the bounding box and appearance specific weights, respectively, at the measurement locations. The two sets of similarity maps discussed earlier are weighted using these weight maps to produce a consolidated set of similarity maps. Finally, the convolved output of the map will now have the similarity scores between the corresponding target with every measurement as the features are normalized to unit vectors by the network branches. Similarity maps are only built during inference. In training, the dot product of the target and measurement unit feature vectors is directly fed to the loss function. Finally, additional $N_{max} - N$ channels are added to the similarity maps as dummy maps (with zeros) for consistency. The result is the local Similarity map with dimensions $21 \times 21 \times N_{max}$.

E. SimNet Loss Function

The SimNet is a mixed architecture of trainable and non-trainable components. The bounding box branch, appearance branch, and importance branches have trainable components, which need a loss function during training to measure the deviation of the prediction from the ground truth. As the problem is defined as finding the similarity score between a target and a measurement, we use weighted cosine distance as the loss function. Cosine distance is widely used as a metric to measure the similarity or dissimilarity between two elements, especially in natural language processing models. The training examples gener-

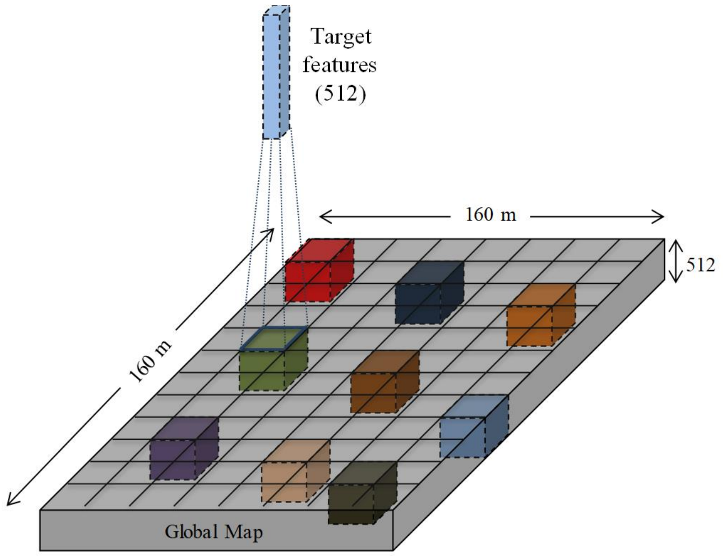


Figure 3.6: Construction of local similarity map for a target. Local similarity maps are built only during inference.

ated from the training dataset have some skewness with respect to the number of positive and negative examples (similar and dissimilar pairs). To account for this skewness, we use the weighted cosine distance. The loss function for the learnable parameters Θ_1 is given by

$$L(\Theta) = \frac{1}{N^+} \sum\nolimits_{i=1}^{N} w_{\text{skew}}^{(i)} \times w_{\text{cost}}^{(i)} \times \big(1 - y^{(i)} \times \hat{y}^{(i)}(\Theta)\big) \tag{3.3}$$

where N^+ is the number of examples with nonzero weights, $y^{(i)}$ denotes the ground truth value of the i^{th} example, i.e., $y^{(i)} \in \{-1, 1\}$. $\hat{y}^{(i)}$ is the estimated cosine similarity score computed using the cosine similarities from the two branches and their normalized importance weights as follows:

$$\hat{y}^{(i)}(\Theta) = \omega_{bbox}(\Theta)^{(i)} \times \hat{y}_{bbox}^{(i)}(\Theta) + \omega_{appear}^{(i)}(\Theta) \times \hat{y}_{appear}^{(i)}(\Theta) \tag{3.4}$$

$w_{\text{skew}}^{(i)}$ is the weight used to remove the imbalance of negative examples in the training dataset. It is based on the ratio of positive and negative examples (18:25), and is given by

$$w_{\text{skew}}^{(i)} = \begin{cases} 18/43 \, if \; y^{(i)} = 1, \\ 25/43 \, if \; y^{(i)} = -1, \end{cases} \tag{3.5}$$

$w_{\text{cost}}^{(i)}$ is given by

$$w_{\text{cost}}^{(i)} = \begin{cases} -\log\left(1 - \cos^{-1}\left(\hat{y}^{(i)}\left(\Theta\right)\right)/\pi + \epsilon\right) \; if \; y^{(i)} = 1, \\ -\log\left(\cos^{-1}\left(\hat{y}^{(i)}\left(\Theta\right)\right)/\pi + \epsilon\right) \; if \; y^{(i)} = -1, \end{cases} \tag{3.6}$$

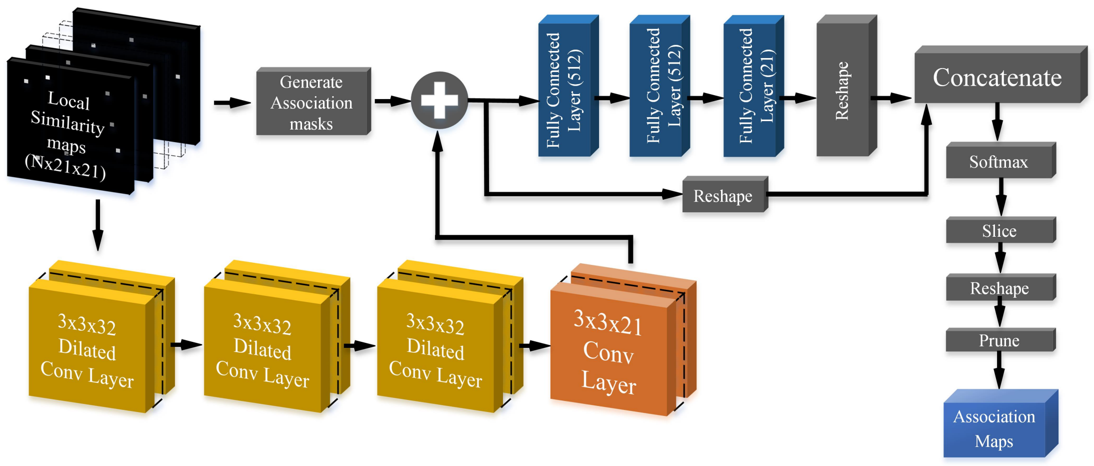
scales the loss function according to how easy or hard it is to distinguish between each pair of examples, so that the training can revolve around a sparse set of the selected hard examples [94]. In Eq. (3.6), ϵ is a small constant $(1e-10)$ that prevents taking log of zero.

3.2.2 AssocNet: Data Association Network

The second network in the architecture is AssocNet. It performs the actual data association between the targets and measurements using the similarity scores provided by the SimNet. Fig. 3.7 gives an overview of AssocNet. The input to this network is a set of local similarity maps from SimNet, having dimensions $21 \times 21 \times N_{max}$. The *outputs* from the network are the probability distributions for each target of being associated to any measurement or to none.

We now discuss the data flow through *AssocNet*. The input local similarity maps are fed through a series of dilated convolutional layers [54] with dilation factors 2, 4, and 6. The neighbouring fields have slightly overlapping fields of view due to increased dilation size [135]. The final convolutional layer enables interactions between these neighbouring units, which effectively results in considering all the detections simultaneously while making assignments. Thus to aggregate information, we use a 3×3 convolutional layer at the end to compute the maps of logits (the vector of non-normalized predictions).

AssocNet is trained to predict assignment probabilities between a target and its probable detections. Since the locations of probable detections are known in each local similarity map, there is no need to train *AssocNet* to predict assignment probabilities of other locations as zero. This reduces the training efforts for regions that are not measurement locations, thereby helping in faster convergence. To implement this idea, we generate association masks for each local similarity map. In the association masks, cells of probable detections are set to zero, while the other cells are set to the smallest floating point negative number (approximating $-\infty$). This ensures the locations that do not contain the



Local Similarity maps (Nx21x21)
Generate Association masks
Fully Connected Layer (512)
Fully Connected Layer (512)
Fully Connected Layer (21)
Reshape
Concatenate
Reshape
Softmax
Slice
Reshape
Prune
Association Maps
3x3x32 Dilated Conv Layer
3x3x32 Dilated Conv Layer
3x3x32 Dilated Conv Layer
3x3x21 Conv Layer

Figure 3.7: Detailed architecture of Assocnet

measurements remain blocked. Then the association masks are added to the map of logits obtained from the convolutional layer with $3 \times 3 \times N_{max}$ filters (see Fig. 3.7). This masking approach maintains the values of the logits computed for probable detections, but makes other logits insignificant for further computation.

After masking the maps of logits, AssocNet splits into two branches. The first branch has two fully connected layers with 512 neurons each and then another fully connected layer with N_{max} neurons. The output of this branch predicts the N_{max} logit values of spurious detections. These are the probabilities that the corresponding target has gone un-detected in the current frame, which also means that no association with any of the measurements could be possible. The output from this branch thus has a shape of $[1, N_{max}]$ representing one value of probability for each of the N_{max} (we consider N_{max} instead of N for consistency and for ease of implementation) targets. The second branch flattens the maps by maintaining the channels dimension such that the shape of the second branch would be $[21 \times 21, N_{max}]$.

The first dimension of size 21×21 corresponds to the logits for associating the target to all possible measurements in the spatial neighborhood of the target. The network itself does not handle duplicate assignments. For example, the same measurement could be assigned to two different targets. We handle this later by removing duplicate assignments based on the association score. If there is a tie, we choose an assignment in random. Thus, finally we solve the data association problem by modeling it as a classification problem. Further we concatenate the two branches along the first dimensions such that the final shape would be $[21 \times 21 + 1, N_{max}]$. Here we add another class to the existing 21×21 classes, which will correspond to the scenario where the target has gone undetected, for example as occlusion. Finally a softmax is applied to this tensor to compute the probabilities of the classes for each target. The resulting tensor is sliced to obtain the class probabilities for the 21×21 measurement locations and the 'occluded' class separately. The probabilities are reshaped to $[21, 21, N_{max}]$, and the last unused channels are pruned, resulting in the final association maps. The probabilities of occlusions are also pruned to remove the unused target channels and to obtain the final vector with the probabilities of occlusions. From the association maps we can obtain the associated measurements for every target by computing the max index locations where the probabilities are maximum.

$$(x^i_{pred}, y^i_{pred}) = \underset{x \in X^i, y \in Y^i}{\operatorname{argmax}}\, p(x, y) \qquad \forall i \in N \tag{3.7}$$

where X^i Y^i correspond to the spatial indices of the map for a given target channel i and (x^i_{pred}, y^i_{pred}) gives the predicted location of the target for the current frame which will coincide with the measurement in that location. By maintaining a dictionary of the measurement identities and their corresponding locations, we could easily obtain the measurement corresponding to the predicted location. The targets that were not associated to any of the measurements can be handled by using predictions from state estimation algorithm, discussed in the track management section.

A. AssocNet Loss Function

Training the AssocNet is a classification problem in which the labels are the association maps showing the true associations for every target. For this purpose, we use the following loss function:'

$$L(\Theta) = l(\Theta)_{assoc} + l(\Theta)_{occ} + l(\Theta)_{reg} \tag{3.8}$$

where Θ is the set parameters of the association network, and $l(\Theta)_{reg}$ is the $L2$ regularization loss. $l(\Theta)_{assoc}$ is the binary cross-entropy computed for the association maps as follows:

$$\begin{aligned} q_{vec} &= q^{(t)}_{assoc}\left(i,j\right) \times \log\left(\hat{q}^{(t)}_{assoc}\left(i,j;\Theta\right) + \epsilon\right) \\ p_{vec} &= p^{(t)}_{assoc}\left(i,j\right) \times \log\left(\hat{p}^{(t)}_{assoc}\left(i,j;\Theta\right) + \epsilon\right) \\ l\left(\Theta\right)_{assoc} &= \sum\nolimits_{t=1}^{N} \sum\nolimits_{i,j=1}^{21} \left(-q_{vec}\right) + \left(-p_{vec}\right) \end{aligned} \tag{3.9}$$

where $q^{(t)}_{assoc}\left(i,j\right) = 1 - p^{(t)}_{assoc}\left(i,j\right)$ and ϵ is a very small number to avoid taking log of 0. The loss term for probability of occlusion $p^{(t)}_{occ}$ is given by

$$l\left(\Theta\right)_{occ} = \sum\nolimits_{t=1}^{N} \left[-q^{(t)}_{occ} \log\left(\hat{q}^{(t)}_{occ}\left(\Theta\right)\right) - \; p^{(t)}_{occ} \log\left(\hat{p}^{(t)}_{occ}\left(\Theta\right)\right)\right] \tag{3.10}$$

3.2.3 Track Management

The track management module takes care of state estimation, initiation, update, and termination of tracks. We use a Kalman filter for motion prediction and state estimation. We initiate, update and prune tracks with a Bayesian estimation model as specified in [119] with a probability of existence P_e. Our complete tracking algorithm is described in Algorithm 1.

3.3 Experiments

This section details the datasets employed for network training, outlines our experimental procedures, and presents both quantitative and qualitative results from our study.

3.3.1 Dataset

The KITTI Tracking benchmark dataset serves as the basis for both training and evaluation in our study. This dataset is composed of 21 training sequences and 29 test sequences. Given the varied levels of difficulty, occlusion, and clutter in the training sequences, we partition each sequence into an 80% training set and a 20% validation set. This division ensures a balanced representation of the different scenarios in both training and validation sets, mitigating dataset skewness. When training SimNet, we create a training dataset from the training sequences, crafting both positive and negative examples in consecutive frames,

Algorithm 1 Tracker Algorithm

```
1: $P_{surv} = 0.60$                                   ▷ Probability of Survival
2: $\theta_{ex} = 0.40$                                ▷ Existence threshold
3: while true do
4:     Get measurements $m^k$ at time $\tau = k$
5:     if $k = 0$ then
6:         for each $m_i^0$ do
7:             Create new track i
8:         end for
9:     else
10:        PredictTracks:
11:        Perform Kalman Filter Prediction
12:        Compute prior $P_{e_i}^k$
13:        DataAssociation:
14:        Data Association for $t^k$ and $m^k$
15:        UpdateTracks:
16:        Perform Kalman Filter Update
17:        Update $P_{e_i}^k$
18:        For all $(None, m_j^k)$ Create a new track
19:        for all $i \in T$ do
20:            if $P_{e_i}^k < \theta_{ex}$ then
21:                Prune $i$
22:            end if
23:        end for
24:    end if
25: end while
```

as well as in odd and even frames, guided by ground truth data. To simulate detector noise, we implement geometric transformations—translation, rotation, and scaling—on the ground-truth bounding box parameters. The resulting training set has a negative to positive ratio of approximately 18 : 25, providing a large, diverse sample for our model.

The object detector was trained on a composite dataset derived from the KITTI 3D object detection dataset and the 80% segment of the previously mentioned KITTI training dataset, with an initial pre-training phase on a synthetic dataset [67]. This pre-training with the synthetic dataset augments the detection accuracy by approximately 3%. The most effective checkpoint for the object detector was selected based on the highest 3D object detection AP (Average Precision) obtained on the combined dataset's validation set. We plot the P-R curve as shown in Fig. 3.8 and find the best threshold corresponding to the maximum F1 score as 0.28.

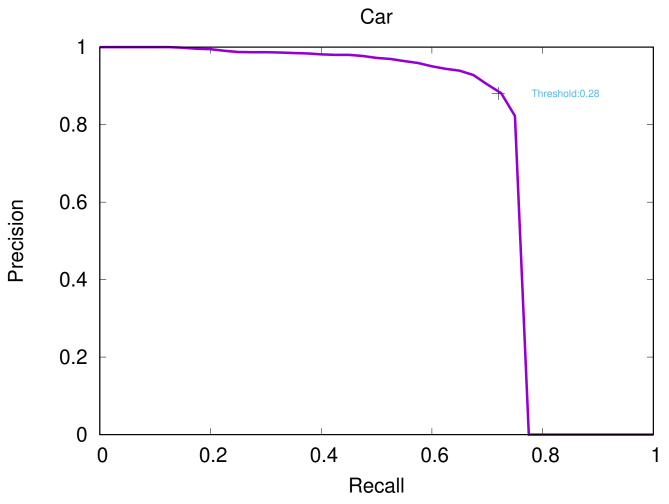


Figure 3.8: PR curve of the object detector (AVOD) on the combined dataset for the car class. The best threshold 0.28 corresponds to the point where F1 score is maximum.

3.3.2 Training Parameters

A. SimNet

SimNet is trained with mini-batches of size 128. Each mini-batch consists of the spatial indices of detections in the global map, the number of targets (N), target centroids in x-y coordinates, target and detection appearance features, their bounding box parameters, and the labels of each example. To optimize the loss function Eq. (3.3) we used Adam optimizer and exponentially-decaying learning rate [75]. The learning rate is initially set to $1e-5$ and then decreased every 100 epochs with a base of 0.95.

B.AssocNet

To optimize the loss function in Eq. (3.8), we used Adam optimizer and exponentially-decaying learning rate. The learning rate was initially set to $1e-6$ and then decreased every 20 epochs with a base of 0.95.

3.3.3 Evaluation Metrics

The accuracy in SimNet is measured Eq. (3.11) by counting the number of correct predictions (similar or dissimilar) by comparing them to the ground truth. We obtain a training accuracy of 90.5% and validation accuracy of 91.3%.

$$accuracy_{simnet} = \frac{\sum_{i=1}^{N}(y^i == \hat{y}^i)}{N} \qquad y \in \{0,1\} \tag{3.11}$$

The AssocNet predicts the (x,y) locations of the probable measurement in the local map of every target. Hence its accuracy is given by considering all the correct predictions in all the target channels as shown in Eq. (3.12). We obtain a training accuracy of 99.71 % and validation accuracy of 99.78 %. We compare these spatial predictions with the ground truth according to the following equation:

$$accuracy_{assocnet} = \frac{\sum_{i=1}^{N}(x^i == \hat{x}^i) \wedge (y^i == \hat{y}^i)}{N} \qquad x, y \in [-10, 10] \tag{3.12}$$

where x^i and y^i are the spatial predictions of associated measurement locations for the target i.

While these metrics are used to evaluate the performance of the individual networks, overall performance of a multi-object tracking algorithm is determined with the help of the CLEAR MOT metrics [13]. The performance of the tracking is measured by two important metrics: Multi-Object Tracking Precision (MOTP) and Multi-Object Tracking Accuracy (MOTA). MOTP is given by:

Method	MOTA ↑	MOTP ↑	MT ↑	PT ↑	ML ↓	IDS ↓	FRAG ↓
Euclidean+*AssocNet*	56.16 %	84.84 %	72.22 %	18.51 %	9.25 %	269	320
Manhattan+*AssocNet*	56.75 %	84.83 %	**73.14 %**	17.59 %	9.25 %	265	319
Bhattacharyya+*AssocNet*	56.69 %	84.81 %	72.22 %	18.51 %	9.25 %	256	307
ChiSquare+*AssocNet*	57.17 %	84.81 %	**73.14 %**	18.51 %	**8.33 %**	262	311
SimNet+Hungarian	74.59 %	**84.92 %**	65.74 %	**23.14 %**	11.11 %	26	93
SimNet+*AssocNet*	**76.52 %**	84.81 %	**73.14 %**	17.59 %	9.25 %	**1**	**54**

(↑ denotes higher values are better. ↓ denotes lower values are better)

Table 3.1: Ablation study on KITTI validation set for 'Car' class

$$\text{MOTP} = \frac{\sum_{i,\tau} d_t^i}{\sum_\tau c_\tau} \tag{3.13}$$

and it measures the total position error for the associated hypotheses over all frames averaged by the total number of matches made. Here d_τ^i is the distance between the i^{th} associated hypothesis and the ground truth detection, and c_τ denotes the total number of associations made at time τ. It represents how precise are the position estimations for the matched target-measurement pairs over all the frames, averaged by the total number of associations made. Multi-Object Tracking Accuracy (MOTA) is given by:

$$\text{MOTA} = 1 - \frac{\sum_\tau (fp_\tau + fn_\tau + \text{IDS})}{\sum_\tau g_\tau} \tag{3.14}$$

where fp_τ denotes the number of false positives, fp_τ denotes the number of false negatives, and IDS denotes the number of ID switches. An ID switch is a case when two tracks interchange their IDs when observed at two time instants $\tau - 1$ and τ.

Mostly Tracked (MT), Mostly Lost (ML), and Fragmentation (FRAG) are three additional metrics [87]. Mostly Tracked is given by the proportion of tracks tracked for more than 80% of their lifetime. Mostly Lost is given by the proportion of tracks tracked for less than 20% of their lifetime. Fragmentation denotes the total number of times a ground truth trajectory is interrupted in the tracking result.

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDS ↓	FRAG ↓
MOTBeyondPixels [142]	**84.24** %	**85.73** %	**73.23** %	**2.77** %	468	944
JCSTD [179]	80.57 %	81.81 %	56.77 %	7.38 %	**61**	643
3D-CNN/PMBM [139]	80.39 %	81.26 %	62.77 %	6.15 %	121	613
extraCK [52]	79.99 %	82.46 %	62.15 %	5.54 %	343	938
MDP [177]	76.59 %	82.10 %	52.15 %	13.38 %	130	**387**
FANTrack (Ours)	77.72 %	82.32 %	62.61 %	8.76 %	150	812

Table 3.2: Results on Kitti Test set for 'Car' class

3.4 Results

3.4.1 Ablation Study

We do an ablation study to evaluate the components in our approach by comparing them with traditional approaches. Firstly, we study the impact of the similarity network. In Tab. 3.1, Euclidean and Manhattan denote the baseline distances modeled with the 3D position estimates. Bhattacharyya and ChiSquare metrics are built from the image histograms of the cropped targets and detections to study the image-only configuration. SimNet and AssocNet denote our Similarity and Association networks respectively. From Tab. 3.1, we could infer that conventional similarity approaches were not able to achieve comparable accuracy (MOTA) as the features involved in the computation of the similarity scores were not robust. We also study the impact of our association network by replacing it with a baseline Hungarian approach. Again, we could observe that the baseline approaches like Hungarian couldn't fare better than ours.

3.4.2 Benchmark Results

We evaluate our approach on the test sequences on the KITTI evaluation server for the 'Car' class. The results are presented in Tab. 3.2. Due to the challenging nature of online tracking approach and to do a fair comparison, we only consider published online tracking approaches for our comparison. We achieve competitive results with respect to the state of the art in online tracking with improved MOTP which is better than most of the online methods. Our Mostly Tracked and Mostly Lost (MT & ML) values are also competitive which show the effectiveness of our data association approach. Further, our approach

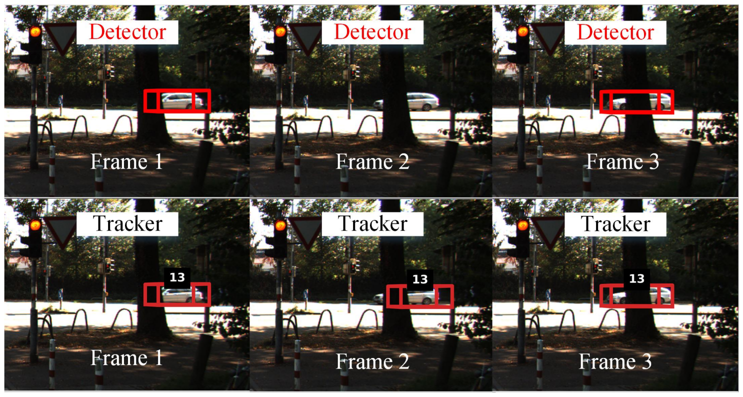


Figure 3.9: Qualitative Evaluation - In this example (video 17 in test set) the detection was missed by the detector and reappears in the next frame. But the tracker was able to successfully maintain the track.

gives inferences in 3D and KITTI evaluations are done in 2D, which is not completely representative of our approach. It should also be noted that none of these approaches use deep learning for data association. On the other side, we have used a simple Kalman filter for state estimation and motion prediction which could potentially be improved by better tuning of parameters or trying out more sophisticated approaches for track management.

After optimizing the convolution operation in Sec. 3.2.1 with selective dot products our tracking algorithm has an average runtime of 0.04s per frame (25 Hz) on Nvidia GeForce GTX 1080 Ti and with a single thread on Intel Core i7-7700 CPU @ 3.60GHz.

3.4.3 Qualitative Evaluation

We perform a qualitative evaluation by running our tracker on the KITTI tracking validation and testing sequences. We analyze different scenarios including occlusions, clutter, parked vehicles and false negatives from the detector. Sec. 3.4.2 shows an example from sequence 14 in the test set. Different tracks representing the vehicles are color coded, and the track IDs are displayed for reference. The tracker is able to perform well in spite of the clutter from the closely parked cars. In Sec. 3.4.2 we see an example from test sequence 17 in which the false negative by the detector is overcome with the help of the prediction of the tracker. These examples show the robustness of the tracker and its ability to per-

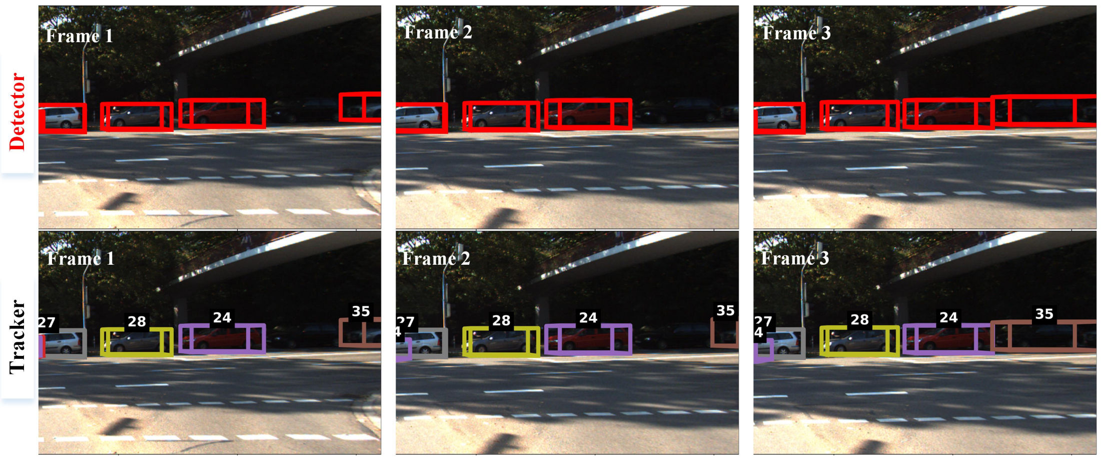


Figure 3.10: Qualitative Evaluation - An example from video 15 in test set where ID switching occurs for Track 35 due to low-lit conditions.

form better in scenarios of missed detections. There were also some cases where the data association fails and as a result ID switching and fragmentation happen. In Sec. 3.4.2 the track 38 was previously assigned to a nearby car, but after a missed detection, ID switching happens. This could be due to the low-lit conditions of the two cars.

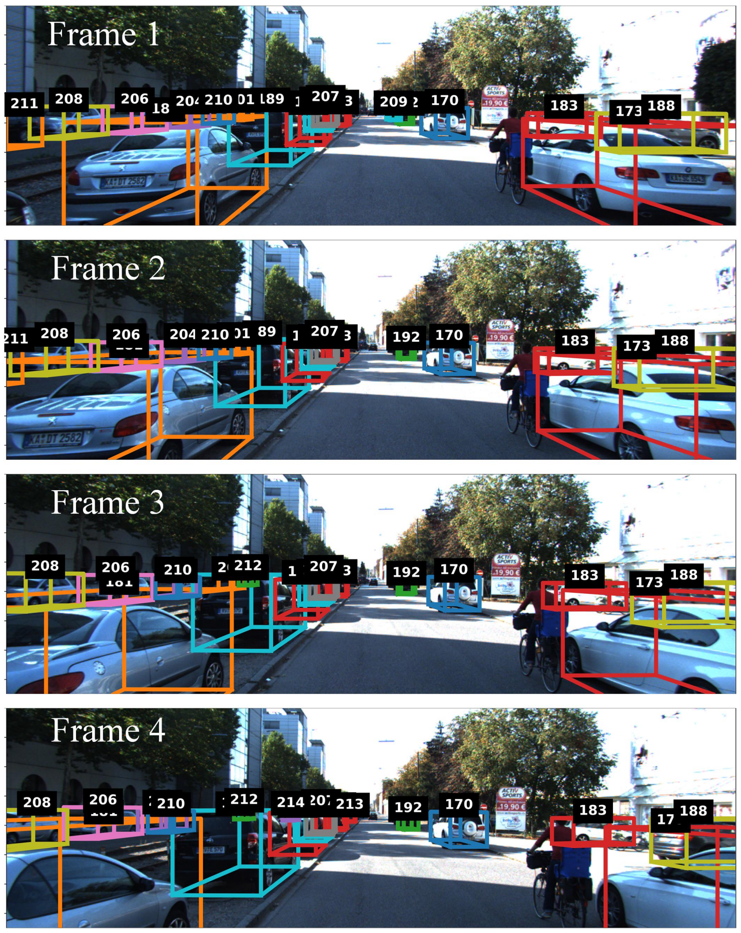


Figure 3.11: Qualitative Evaluation - An example from video 14 in test set where the tracker performs well in a cluttered scene with parked cars.

3.5 Chapter Conclusion

In this chapter, we presented a solution to the problem of data association in 3D online multi-object tracking using deep learning with multi-modal data. We have shown that a learning-based data association framework helps in combining different similarity cues in the data and provides more accurate associations than conventional approaches, which helps in increased overall tracking performance. We demonstrated the effectiveness of the tracker built using this model with a multitude of experiments and evaluations and show competitive results in the KITTI tracking benchmark.

In this chapter, we proposed an improved solution to the data-association problem, without modifying the features of the 3D object detector. Nevertheless, data-association could also be improved through enhancing the features of the 3D object detection itself. Consequently, in the next chapter, Chapter 4, we will investigate a technique designed to boost the efficacy of 3D object detection.

Chapter 4

SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection

Following the exploration of data-association improvement for 3D object tracking in Chapter 3, this chapter focuses on another facet of the problem — improving the quality of object detectors. We hypothesize that improving the quality of object detectors can also enhance data-association performance in object tracking. Accurate and reliable object detectors streamline data association by delivering precise spatial and temporal data, facilitating correspondence across successive frames, thus diminishing ambiguity and mismatches. Further, advanced detectors can more accurately identify occluded or partially visible objects, boosting track continuity and reducing track fragmentation.

To this end, we present two self-attention variants for 3D object detection aimed at augmenting convolutional features with self-attention features, and at explicitly modeling long-range interactions. Moreover, we introduce a technique that samples a subset of the most representative features, permitting the model to scale to larger point-clouds and generate more distinguishing and informative feature descriptors. Our proposed method exhibits versatility, enhancing detection performance when applied to most state-of-the-art detectors. SA-Det3D code and documentation is open-source.[1,2]

[1]https://github.com/AutoVision-cloud/SA-Det3D
[2]https://github.com/AutoVision-cloud/Deformable-PV-RCNN

4.1 Motivation and Contributions

3D object detection has been receiving increasing attention in the computer vision community, driven by the ubiquity of LiDAR sensors and its widespread applications in autonomous driving. Point-cloud based 3D object detection has especially witnessed tremendous advancement in recent years [83, 184, 144, 143, 128, 188, 191, 55, 207, 127]. Grid-based methods first transform the irregular point-clouds to regular representations such as 2D bird's-eye view (BEV) maps or 3D voxels and process them using 2D/3D convolutional networks (CNNs). Point-based methods sample points from the raw point-cloud and query a local group around each sampled point to define convolution-like operations [130, 161, 173] for point-cloud feature extraction. Both 2D/3D CNNs and point-wise convolutions process a local neighbourhood and aggregate global context by applying feature extractors hierarchically across many layers. This has several limitations: the number of parameters scales poorly with increased size of the receptive field; learned filters are stationary across all locations; and it is challenging to coordinate the optimization of parameters across multiple layers to capture patterns in the data [200].

In addition, point-cloud based 3D object detectors have to deal with missing/noisy data and a large imbalance in points for nearby and faraway objects. This motivates the need for a feature extractor that can learn global point-cloud correlations to produce more powerful, discriminative and robust features. For example, there is a strong correlation between the orientation features of cars in the same lane and this can be used to produce more accurate detections especially for distant cars with fewer points. High-confidence false positives produced by a series of points that resemble a part of an object can be also be eliminated by adaptively acquiring context information at increased resolutions.

Self-attention [164] has recently emerged as a basic building block for capturing long-range interactions. The key idea of self-attention is to acquire global information as a weighted summation of features from all positions to a target position, where the corresponding weight is calculated *dynamically* via a similarity function between the features in an embedded space at these positions. The number of parameters is independent of the scale at which self-attention processes long-range interactions. Inspired by this idea, we propose two self-attention based context-aware modules to augment the standard convolutional features—Full Self-Attention (FSA) and Deformable Self-Attention (DSA). Our FSA module computes pairwise interactions among all non-empty 3D entities, and the DSA module scales the operation to large point-clouds by computing self-attention on a representative and informative subset of features. Our experiments show that we can improve the performance of current 3D object detectors with our proposed FSA/DSA blocks while simultaneously promoting parameter and compute efficiency.

Contributions

- We propose a generic globally-adaptive context aggregation module that can be applied across a range of modern architectures including BEV [83], voxel [184], point [144] and point-voxel [143] based 3D detectors. We show that we can outperform strong baseline implementations by up to 1.5 3D AP (average precision) while simultaneously reducing parameter and compute cost by 15-80% and 30-50%, respectively, on the KITTI validation set.
- We design a scalable self-attention variant that learns to deform randomly sampled locations to cover the most representative and informative parts and aggregate context on this subset. This allows us to aggregate global context in large-scale point-clouds like nuScenes and Waymo Open dataset.
- Extensive experiments demonstrate the benefits of our proposed FSA/DSA modules by consistently improving the performance of state-of-the-art detectors on KITTI [43], nuScenes [19] and Waymo Open dataset [156].

4.2 Standard Feature Extractors for 3D Object Detection

In this section, we briefly review the standard feature extractors for 3D object detection to motivate our design. 2D and 3D convolutions have achieved great success in processing pillars [83] and voxel grids [184] for 3D object detection. Point-wise feature learning methods like PointNet++ [130] have also been successful in directly utilizing sparse, irregular points for 3D object detection [144].

Given a set of vectors $\{x_1, x_2, ...x_n\}$, which can represent pillars, voxels or points, with $x_i \in R^C$, one can define a function $f : \mathcal{X} \rightarrow R^{C'}$ that maps them to another vector. In this case, standard convolution at location $\hat{p}$ can be formulated as:

$$f(\hat{p}) = \sum_{l \in \Omega_1} x_{\hat{p}+l} w_l \tag{4.1}$$

where w is a series of C' dimensional weight vectors with kernel size $2m + 1$ and $\Omega_1 = [l \in (-m, ...m)]$ representing the set of positions relative to the kernel center. Similarly, a point-feature approximator at $\hat{p}$ can be formulated as:

$$f(\hat{p}) = \max_{l \in \Omega_2} h(x_l) \tag{4.2}$$

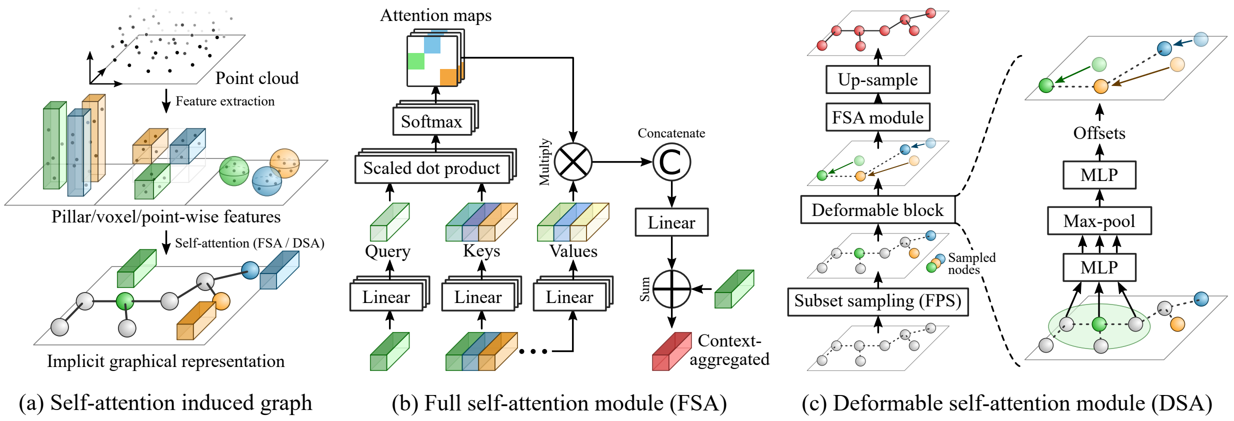


Figure 4.1: Architectures of the proposed FSA and DSA modules.

where h is a C' dimensional fully connected layer, max denotes the max-pooling operator and Ω_2 denotes the k-nearest neighbors of $\hat{p}$. The operator f thus aggregates features with pre-trained weights, h and w, from nearby locations.

Limitations One of the disadvantages of this operator is that weights are fixed and cannot adapt to the content of the features or selectively focus on the salient parts. Moreover, since the number of parameters scales linearly with the size of the neighborhood to be processed, long range feature-dependencies can only be modeled by adding more layers, posing optimization challenges for the network. Since useful information for fine-grained object recognition and localization appears at both global and local levels of a point-cloud, our work looks for more effective feature aggregation mechanisms.

4.3 SA-Det3D

In this section, we first introduce a Full Self-Attention (FSA) module for discriminative feature extraction in 3D object detection that aims to produce more powerful and robust representations by exploiting global context. Next, inspired by 2D deformable convolutions [33] we introduce a variant of FSA called Deformable Self-Attention (DSA). DSA can reduce the quadratic computation time of FSA and scale to larger and denser point-clouds. The two proposed modules are illustrated in Fig. 4.1.

4.3.1 Formulation

For the input set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n\}$ of n correlated features and $i \in \{1, ...n\}$, we propose to use self-attention introduced by Vaswani et al. [164] to exploit the pairwise similarities of the i^{th} feature node with all the feature nodes, and stack them to compactly represent the global structural information for the current feature node.

Mathematically, the set of pillar/voxel/point features and their relations are denoted by a graph $G = (\mathcal{V}, \mathcal{E})$, which comprises the node set $\mathcal{V} = \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n \in R^d\}$, together with an edge set $\mathcal{E} = \{\mathbf{r}_{i,j} \in R^{N_h}, i = 1, ..., n \text{ and } j = 1, ..., n\}$. A self-attention module takes the set of feature nodes, and computes the edges (see Fig. 4.1 (a)). The edge $\mathbf{r}_{i,j}$ represents the relation between the i^{th} node and the j^{th} node, and N_h represents the number of heads (number of attention maps in Fig. 4.1 (b)) in the attention mechanism across d feature input channels as described below. We assume that N_h divides d evenly. The advantage of representing the processed point-cloud features as nodes in a graph is that now the task of aggregating global context is analogous to capturing higher order interaction among nodes by message passing on graphs for which many mechanisms like self-attention exist.

4.3.2 Full Self-Attention Module

Our Full Self-Attention (FSA) module projects the features $\mathbf{x}_i$ through linear layers into matrices of query vectors Q, key vectors K, and value vectors V (see Fig. 4.1(b)). The similarities between query $\mathbf{q}_i$ and all keys, $\mathbf{k}_{j=1:n}$, are computed by a dot-product, and normalized into attention weights $\mathbf{w}_i$, via a softmax function. The attention weights are then used to compute the pairwise interaction terms, $\mathbf{r}_{ij} = w_{ij}\mathbf{v}_j$. The accumulated global context for each node vector $\mathbf{a}_i$ is the sum of these pairwise interactions, $\mathbf{a}_i = \sum_{j=1:n} \mathbf{r}_{ij}$. As we mentioned in our formulation, we also use multiple attention heads, applied in parallel, which can pick up channel dependencies independently. The final output for the node i is then produced by concatenating the accumulated context vectors $\mathbf{a}_i^{h=1:N_h}$ across heads, passing it through a linear layer, normalizing it with group normalization [175] and summing it with $\mathbf{x}_i$ (residual connection).

Advantages: The important advantage of this module is that the resolution at which it gathers context is independent of the number of parameters and the operation is permutation-invariant. This makes it attractive to replace a fraction of the parameter-heavy convolutional filters at the last stages of 3D detectors with self-attention features for improved feature quality and parameter efficiency.

Complexity: The pairwise similarity calculation is $\mathcal{O}(n^2 d)$ in nature. The inherent sparsity of point-clouds and the efficient matrix-multiplication based pairwise computation makes FSA a viable feature extractor in current 3D detection architectures. However, it is necessary to trade accuracy for computational efficiency in order to scale to larger point-clouds. In the next section, we propose our Deformable Self-Attention module to reduce the quadratic computation time of FSA.

4.3.3 Deformable Self-Attention Module

Our primary idea is to attend to a representative subset of the original node vectors in order to aggregate global context. We then up-sample this accumulated structural information back to all node locations. The complexity of this operation is $\mathcal{O}(m^2 d)$, where $m << n$ is the number of points chosen in the subset. In order for the subset to be representative, it is essential to make sure that the selected nodes cover the informative structures and common characteristics in 3D geometric space. Inspired by deformable convolution networks [33] in vision, we propose a geometry-guided vertex refinement module that makes the nodes self-adaptive and spatially recomposes them to cover locations which are important for semantic recognition. Our node offset-prediction module is based on vertex alignment strategy proposed for domain alignment [131, 48]. Initially m nodes are sampled from the point-cloud by farthest point sampling (FPS) with vertex features $\mathbf{x}_i$ and a 3D vertex position v_i. For the i^{th} node, the updated position v'_i is calculated by aggregating the local neighbourhood features with different significance as follows:

$$x_i^* = \frac{1}{k}\text{ReLU} \sum_{j \in \mathcal{N}(i)} W_{\text{offset}}(\mathbf{x}_i - \mathbf{x}_j) \cdot (v_i - v_j) \tag{4.3}$$

$$v'_i = v_i + \tanh(W_{\text{align}} x_i^*) \tag{4.4}$$

where $\mathcal{N}_i$ gives the i-th node's k-neighbors in the point-cloud and W_{offset} and W_{align} are weights learned end-to-end. The final node features are computed by a non-linear processing of the locally aggregated embedding as follows:

$$\mathbf{x}'_i = \max_{j \in \mathcal{N}(i)} W_{out} \mathbf{x}_j \tag{4.5}$$

Next, the m adaptively aggregated features $\{\mathbf{x}'_1 .... \mathbf{x}'_m\}$ are then passed into a full self-attention (FSA) module to model relationships between them. This aggregated global information is then shared among all n nodes from the m representatives via up-sampling. We call this module a Deformable Self-Attention (DSA) module as illustrated in Fig. 4.1(c).

Upsampling for DSA: Given the features for m sampled, deformed and attended points, we explore *two* up-sampling methods to distribute the accumulated structural information back to all n node locations. We first test the feature propagation method proposed in PointNet++ [130] to obtain point features for all the original nodes. This works well for most of our experiments, especially on the KITTI and the Waymo Open Dataset. While this is simple and easy to implement, a draw-back is that the interpolation radius has to be chosen empirically. To avoid choosing an interpolation radius for the diverse classes present in the nuScenes dataset, we explore an attention-based up-sampling method as proposed in [86]. The set of m points is attended to by the n node features to finally produce a set of n elements. This up-sampling method works well for the nuScenes dataset.

Advantages: The main advantage of DSA is that it can scalably aggregate global context for pillar/voxel/points. Another advantage of DSA is that it is trained to collect information from the most informative regions of the point-cloud, improving the feature descriptors.

4.4 Experiments

In this section, we provide details regarding the implementation of SA-Det3D and the dataset used in our study.

4.4.1 Network Architectures

We train and evaluate our proposed FSA and DSA modules on four state-of-the-art architecture backbones: PointPillars [83], SECOND [184], Point-RCNN [144], and PV-RCNN [143]. The architectures of the backbones are illustrated in Figure 4.2. The augmented backbones can be trained end-to-end without additional supervision.
For the KITTI dataset, the detection range is within [0,70.4] m, [-40,40] m and [-3,1] m for the XYZ axes, and we set the XY pillar resolution to (0.16, 0.16) m and XYZ voxel-resolution of (0.05, 0.05, 0.1) m. For nuScenes, the range is [-50,50] m, [-50,50] m, [-5,3] m along the XYZ axes and the XY pillar resolution is (0.2, 0.2) m. For the Waymo Open dataset, the detection range is [-75.2, 75.2] m for the X and Y axes and [-2, 4] m for the Z-axis, and we set the voxel size to (0.1, 0.1, 0.15) m. Additionally, the deformation radius is set to 3 m, and the feature interpolation radius is set to 1.6 m with 16 samples. The self-attention feature dimension is 64 across all models. We apply 2 FSA/DSA modules

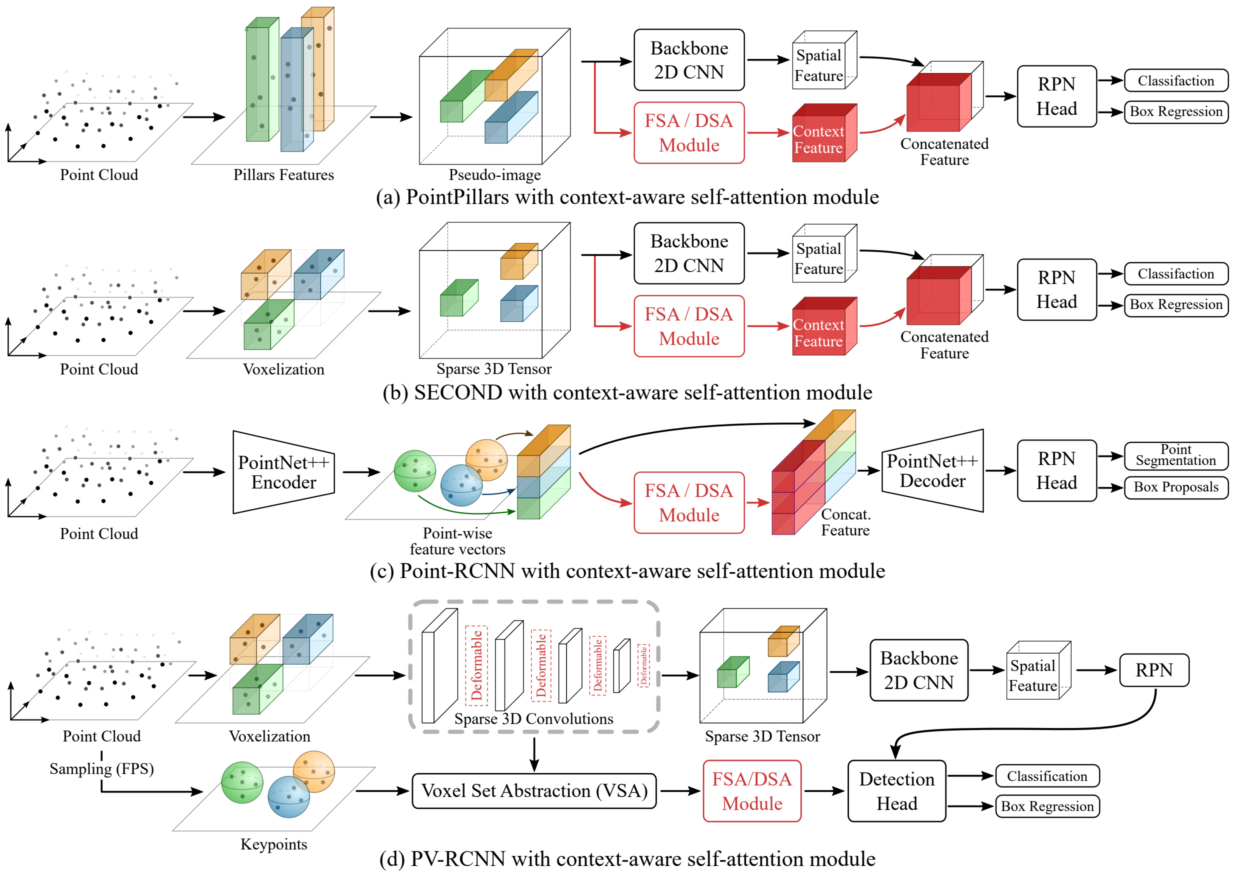


Figure 4.2: Proposed FSA/DSA module augmented network architectures for different backbone networks.

with 4 attention heads across our chosen baselines. For DSA, we use a subset of 2,048 sampled points for KITTI and 4,096 sampled points for nuScenes and Waymo Open Dataset. We use standard data-augmentation for point clouds. For baseline models, we reuse the pre-trained checkpoints provided by OpenPCDet [160]. More implementation details are presented in the Appendix.

4.4.2 Implementation Details

KITTI: KITTI benchmark [43] is a widely used benchmark with 7,481 training samples and 7,518 testing samples. We follow the standard split [29] and divide the training samples into *train* and *val* split with 3,712 and 3,769 samples respectively. All models were trained

Method	PointPillars [83]				SECOND [184]			
	3D	BEV	Param	FLOPs	3D	BEV	Param	FLOPs
Baseline	78.39	88.06	4.8 M	63.4 G	81.61	88.55	4.6 M	76.9 G
DSA	78.94	88.39	1.1 M	32.4 G	**82.03**	89.82	2.2 M	52.6 G
FSA	**79.04**	**88.47**	1.0 M	31.7 G	81.86	**90.01**	2.2 M	51.9 G
Improve.	**+0.65**	**+0.41**	**-79%**	**-50%**	**+0.42**	**+1.46**	**-52%**	**-32%**

Method	Point-RCNN [144]				PV-RCNN [143]			
	3D	BEV	Param	FLOPs	3D	BEV	Param	FLOPs
Baseline	80.52	**88.80**	4.0 M	27.4 G	84.83	**91.11**	12 M	89 G
DSA	81.80	88.14	2.3 M	19.3 G	84.71	90.72	10 M	64 G
FSA	**82.10**	88.37	2.5 M	19.8 G	**84.95**	90.92	10 M	64.3 G
Improve.	**+1.58**	-	**-37%**	**-38%**	**+0.12**	-	**-16%**	**-27%**

Table 4.1: Performance comparison for moderate difficulty Car class on KITTI *val* split with 40 recall positions

on 4 NVIDIA Tesla V100 GPUs for 80 epochs with Adam optimizer [75] and one cycle learning rate schedule [150]. We also use the same batch size and learning rates as the baseline models.

nuScenes nuScenes [19] is a more recent large-scale benchmark for 3D object detection. In total, there are 28k, 6k, 6k, annotated frames for training, validation, and testing, respectively. The annotations include 10 classes with a long-tail distribution. We train and evaluate a DSA model with PointPillars as the backbone architecture. All previous methods combine points from current frame and previous frames within 0.5 s, gathering about 300 k points per frame. FSA does not work in this case since the number of pillars in a point cloud is too large to fit the model in memory. In DSA, this issue is avoided by sampling a representative subset of pillars. The model was trained on 4 NVIDIA Tesla V100 GPUs for 20 epochs with a batch size of 8 using Adam optimizer [75] and one cycle learning rate schedule [150].

Waymo Open Dataset Waymo Open Dataset [156] is currently the largest dataset for 3D detection for autonomous driving. There are 798 training sequences with 158,081 LiDAR samples, and 202 validation sequences with 39,987 LiDAR samples. The objects are annotated in the full 360°field of view. We train and evaluate a DSA model with SECOND as the backbone architecture. The model was trained on 4 NVIDIA Tesla V100 GPUs for 50 epochs with a batch size of 8 using Adam optimizer [75] and one cycle learning rate schedule [150].

Model	Car - 3D			Car - BEV			Cyclist - 3D			Cyclist - BEV		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [29]	74.97	63.63	54.00	86.62	78.93	69.80	-	-	-	-	-	-
PointPillars [83]	82.58	74.31	68.99	90.07	86.56	82.81	77.10	58.65	51.92	79.90	62.73	55.58
SECOND [184]	83.34	72.55	65.82	89.39	83.77	78.59	71.33	52.08	45.83	76.50	56.05	49.45
PointRCNN [144]	86.96	75.64	70.70	92.13	87.39	82.72	74.96	58.82	52.53	82.56	67.24	60.28
STD [191]	87.95	79.71	75.09	94.74	89.19	**86.42**	78.69	61.59	55.30	81.36	67.23	59.35
3DSSD [188]	88.36	79.57	74.55	92.66	89.02	85.86	**82.48**	64.10	56.90	**85.04**	67.62	61.14
SA-SSD [55]	88.75	79.79	74.16	**95.03**	**91.03**	85.96	-	-	-	-	-	-
TANet [100]	83.81	75.38	67.66	-	-	-	73.84	59.86	53.46	-	-	-
Point-GNN [146]	88.33	79.47	72.29	93.11	89.17	83.90	78.60	63.48	57.08	81.17	67.28	59.67
PV-RCNN [143]	**90.25**	81.43	76.82	94.98	90.65	86.14	78.60	63.71	57.65	82.49	68.89	62.41
PV-RCNN + DSA (Ours)	88.25	**81.46**	**76.96**	92.42	90.13	85.93	82.19	**68.54**	**61.33**	83.93	**72.61**	**65.82**

Table 4.2: Performance comparison of 3D detection on KITTI *test* split with AP calculated with 40 recall positions. The **best** and second-best performances are highlighted across all datasets.

4.5 Results

4.5.1 3D Detection on the KITTI Dataset

On KITTI, we report the performance of our proposed model on both *val* and *test* split. We focus on the average precision for moderate difficulty and two classes: car and cyclist. We calculate the average precision on *val* split with 40 recall positions using IoU threshold of 0.7 for car class and 0.5 for cyclist class. The performance on *test* split is calculated using the official KITTI test server. **Comparison with state-of-the-art:** Tab. 4.1 shows the results for car class on KITTI *val* split. For all four state-of-the-art models augmented with DSA and FSA, both variants were able to achieve performance improvements over strong baselines with significantly fewer parameters and FLOPs. On KITTI *test* split, we evaluate PV-RCNN+DSA and compare it with the models on KITTI benchmark. The results are shown in Tab. 4.2. On the car class DSA shows an improvement of 0.15 3D AP on the hard setting, while for the smaller cyclist class we achieve significantly better performance than all other methods with upto 4.5 3D AP improvement on the moderate setting. Overall, the results consistently demonstrate that adding global contextual information benefits performance and efficiency, especially for the difficult cases with smaller number of points.

Model	Mode	mAP	NDS	Car	Truck	Bus	Trailer	CV	Ped	Moto	Bike	Tr. Cone	Barrier
PointPillars [83]	Lidar	30.5	45.3	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
WYSIWYG [63]	Lidar	35.0	41.9	79.1	30.4	46.6	40.1	7.1	65.0	18.2	0.1	28.8	34.7
PointPillars+ [165]	Lidar	40.1	55.0	76.0	31.0	32.1	36.6	11.3	64.0	34.2	14.0	45.6	56.4
PMPNet [192]	Lidar	45.4	53.1	79.7	33.6	47.1	43.0	**18.1**	**76.5**	40.7	7.9	58.8	48.8
SSN [209]	Lidar	46.3	56.9	80.7	37.5	39.9	43.9	14.6	72.3	**43.7**	20.1	54.2	56.3
Point-Painting [165]	RGB + Lidar	46.4	58.1	77.9	35.8	36.2	37.3	15.8	73.3	41.5	**24.1**	**62.4**	**60.2**
PointPillars + DSA (Ours)	Lidar	**47.0**	**59.2**	**81.2**	**43.8**	**57.2**	**47.8**	11.3	73.3	32.1	7.9	60.6	55.3

Table 4.3: Performance comparison of 3D detection with PointPillars backbone on nuScenes *test* split. “CV”, ”Ped” , “Moto”, “Bike”, “Tr. Cone” indicate construction vehicle, pedestrian, motorcycle, bicycle and traffic cone respectively. The values are taken from the official evaluation server `https://eval.ai/web/challenges/challenge-page/356/leaderboard/1012`.

4.5.2 3D Detection on the nuScenes Dataset

To test the performance of our methods in more challenging scenarios, we evaluate PointPillars with DSA modules on the nuScenes benchmark using the official test server. In addition to average precision (AP) for each class, nuScenes benchmark introduces a new metric called nuScenes Detection Score (NDS). It is defined as a weighted sum between mean average precision (mAP), mean average errors of location (mATE), size (mASE), orientation (mAOE), attribute (mAAE) and velocity (mAVE).

Comparison with state-of-the-art: We first compare our PointPillars+DSA model with PointPillars+ [165], a class-balanced re-sampled version of PointPillars inspired by [208]. DSA achieves about 7% improvement in mAP and 4.2% improvement in NDS compared to PointPillars+, even for some small objects, such as pedestrian and traffic cone. Compared with other attention and fusion-based methods like PMPNet and Point-Painting, DSA performs better in the main categories of traffic scenarios such as Car, Truck, Bus and Trailer etc. Overall, our model has the highest mAP and NDS score compared to state-of-the-art PointPillars-based 3D detectors.

4.5.3 3D Detection on the Waymo Open Dataset

We also report performance on the large Waymo Open Dataset with our SECOND+DSA model to further validate its effectiveness. The objects in the dataset are split into two levels based on the number of points in a single object, where LEVEL1 objects have at-least 5 points and the LEVEL2 objects have at-least 1 point inside. For evaluation, the average precision (AP) and average precision weighted by heading (APH) metrics are used. The IoU threshold is 0.7 for vehicles.

Difficulty	Method	Vehicle	
		3D AP	3D APH
L1	StarNet [110]	53.7	-
	PointPillars [83]	56.6	-
	PPBA [30]	62.4	-
	MVF [29]	62.9	-
	AFDet [42]	63.7	-
	CVCNet [28]	65.2	-
	Pillar-OD [170]	69.8	-
	†SECOND [184]	70.2	69.7
	PV-RCNN [143]	70.3	69.7
	SECOND + DSA (Ours)	**71.1**	**70.7**
L2	†SECOND [184]	62.5	62.0
	PV-RCNN [143]	**65.4**	**64.8**
	SECOND + DSA (Ours)	63.4	63.0

Table 4.4: Comparison on Waymo Open Dataset *validation* split for 3D vehicle detection. Our DSA model has 52% fewer parameters and 32% fewer FLOPs compared to SECOND and 80% fewer parameters and 41% fewer FLOPs compared to PV-RCNN. †Re-implemented by [160]

Comparison with the state-of-the-art: Tab. 4.4 shows that our method outperforms previous state-of-the-art PV-RCNN with a 0.8%AP and 1%APH gain for 3D object detection while having 80% fewer parameters and 41% fewer FLOPs on LEVEL1. This supports that our proposed DSA is able to effectively capture global contextual information for improving 3D detection performance. Better performance in terms of APH also indicates that context helps to predict more accurate heading direction for the vehicles. On LEVEL2, we outperform the SECOND baseline by 0.9% AP and 1.0% APH. Overall SECOND+DSA provides the better balance between performance and efficiency as compared to PV-RCNN. The experimental results validate the generalization ability of FSA/DSA on various datasets.

4.5.4 Ablation studies and qualitative analysis

Ablation studies are conducted on the KITTI validation split [29] for moderate Car class using AP@R40, in order to validate our design choices.

Model variations In our ablation study with PointPillars backbone in Tab. 4.5, we represent the number of 2D convolution filters as $N_{filters}$, self-attention heads as N_h, self-

Model	$N_{filters}$	N_h	N_l	N_{keypts}	r_{def}	r_{up}	3D AP	Params	FLOPs
baseline	(64,128,256)	-	-	-	-	-	78.39	4.8M	63.4G
	(64,64,128)	-	-	-	-	-	78.07	1.5M	31.5G
(A)	(64,64,64)	2	2	-	-	-	78.67	1.0M	31.3G
		4	1	-	-	-	78.34	1.0M	31.5G
		4	2	-	-	-	79.04	1.0M	31.7G
		4	4	-	-	-	78.56	1.0M	32.0G
(B)	(64,64,64)	4	2	512	3	1.6	78.70	1.1M	32.4G
				1024			78.95	1.1M	32.4G
				2048			78.94	1.1M	32.4G
				4096			78.90	1.1M	32.4G
(C)	(64,64,64)	4	2	2048	2	1.6	78.93	1.1M	32.4G
					1.4	1.6	78.22	1.1M	32.4G
					3	2	78.10	1.1M	32.4G
					3	1	78.96	1.1M	32.4G
(D)	(64,128,256)	4	2	2048	2	1	79.80	5.1M	73.5G

Table 4.5: Ablation of model components with PointPillars backbone on KITTI moderate Car class of *val* split.

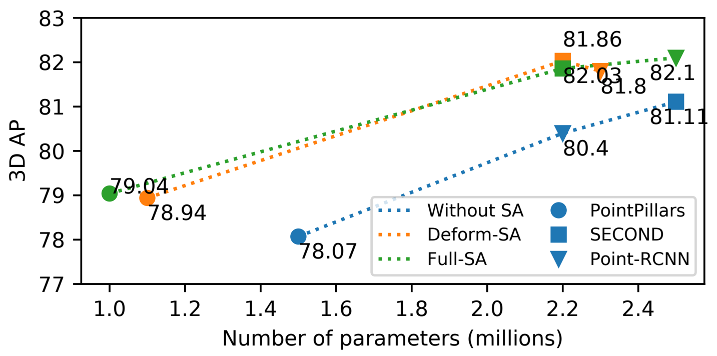
attention layers as N_l, sampled points for DSA as N_{keypts}, deformation radius as r_{def} and the up-sampling radius as r_{up}.

Effect of number of filters: We note that both FSA and DSA outperform not only the models with similar parameters by 0.97% and 0.87% respectively, but also the state-of-the-art models with 80% more parameters by 0.65% and 0.55%. This indicates that our modules are extremely parameter efficient. Finally, we also note that if the number of parameters and compute are kept roughly the same as the baseline(Row-D), DSA outperforms the baseline by a large margin of 1.41%. We also illustrate consistent gains in parameter and computation budget across backbones in Fig. 4.3.

Effect of number of self-attention heads and layers (Row-A): We note that increasing heads from 2 to 4 leads to an improvement of 0.37% for PointPillars. Since increasing number of self-attention layers beyond a certain value can lead to over-smoothing [133], we use 2 FSA/DSA layers in the backbone and 4 heads for multi-head attention.

Effect of number of sampled points (Row-B): For DSA, we also vary the number of keypoints sampled for computation of global context. We note that the performance is relatively robust to the number of sampled points.

Effect of deformation and upsampling radius (Row-C): For DSA, we note that the performance is generally robust to the deformation radius upto a certain threshold, but the up-sampling radius needs to be tuned carefully. Generally an up-sampling radius



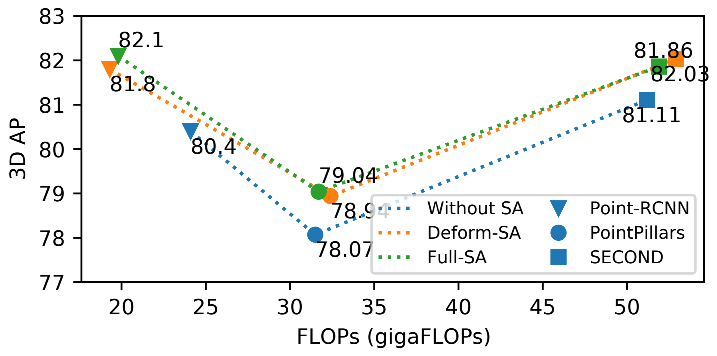


Figure 4.3: 3D AP on moderate Car class of KITTI val split (R40) vs. number of parameters (Top) and GFLOPs (Bottom) for baseline models and proposed baseline extensions with Deformable and Full SA.

of 1.6m in cars empirically works well.

Effect of noise on performance We introduce noise points to each object similar to TANet [100], to probe the robustness of representations learned. As shown in Fig. 4.4, self-attention-augmented models are more robust to noise than the baseline. For example, with 100 noise points added, the performance of SECOND and Point-RCNN drops by 3.3% and 5.7% respectively as compared to SECOND-DSA and Point-RCNN-DSA, which suffer a lower drop of 2.7% and 5.1% respectively.

Effect of number of object points on performance We sort the cars based on the numbers of points in them in increasing order, and divide them into 3 groups based on the sorted order. Then we calculate the 3D AP across every group. As shown in Fig. 4.5, the effect of the self-attention module becomes apparent as the number of points on the cars decreases. For objects with very few points, FSA can increase the 3D AP for PointPillars by 2.8% and PV-RCNN by 1.5%.

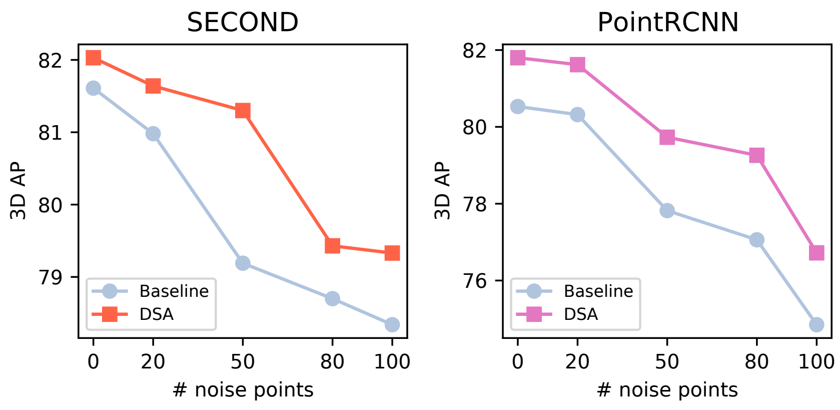


Figure 4.4: 3D AP of SECOND-DSA (orange) & Point-RCNN-DSA (violet) vs. SECOND & Point-RCNN baseline (light-steel-blue) for noise-points per ground-truth bounding box, varying from 0 to 100 on KITTI *val* moderate

Qualitative results In Fig. 4.6, we first show that our FSA-based detector identifies missed detections and eliminates false positives across challenging scenes for different backbones. Next, we identify objects for which addition of self-attention shows the largest increase in detection confidences as shown in Fig. 4.7(a). We then copy-paste the point-clouds for these cars into different scenes. Our expectation is that the FSA is a more robust detector and can detect these examples even when randomly transplanted to different scenes. The first two rows of Fig. 4.7(b) show that FSA is capable of detecting the copy-pasted car in different scenes while the baseline consistently misses them. This supports our motivation that adding contextual self-attention features to convolutional maps results in a more accurate and robust feature extractor. In the third row of Fig. 4.7(b), we show cases for Point-Pillars and PV-RCNN where the orientation is flipped for our FSA-based detector even though the detection confidence remains high. We expect that this confusion occurs because FSA aggregates context from nearby high-confidence detections thereby correlating their orientations.

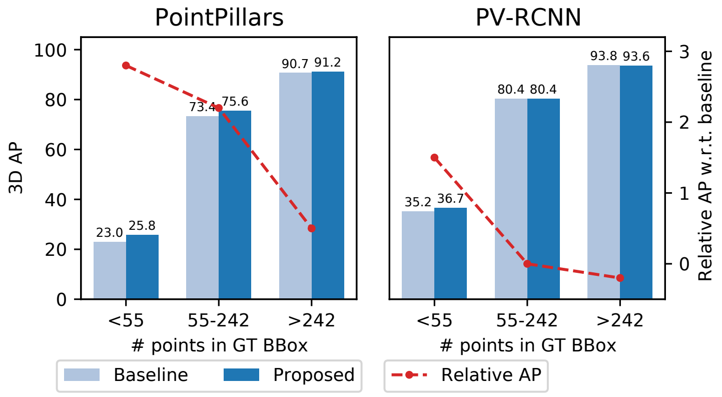


Figure 4.5: 3D AP of PointPillars-FSA, PV-RCNN-FSA and respective baselines vs.number of points in the ground-truth bounding box on KITTI *val*

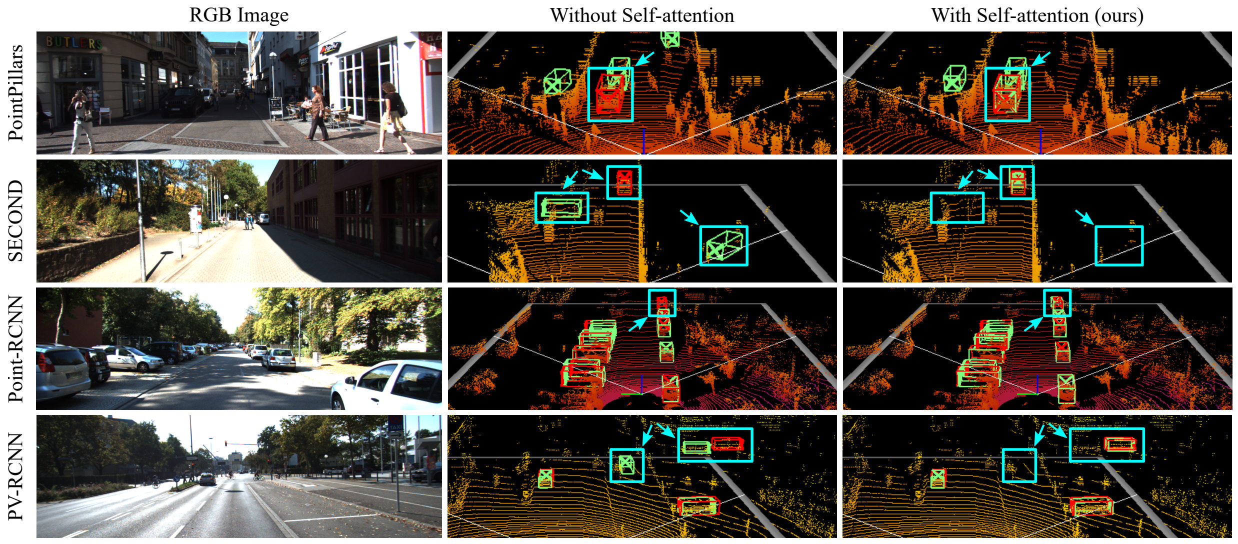
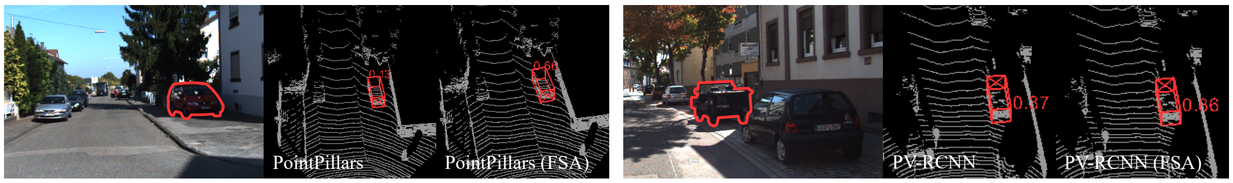
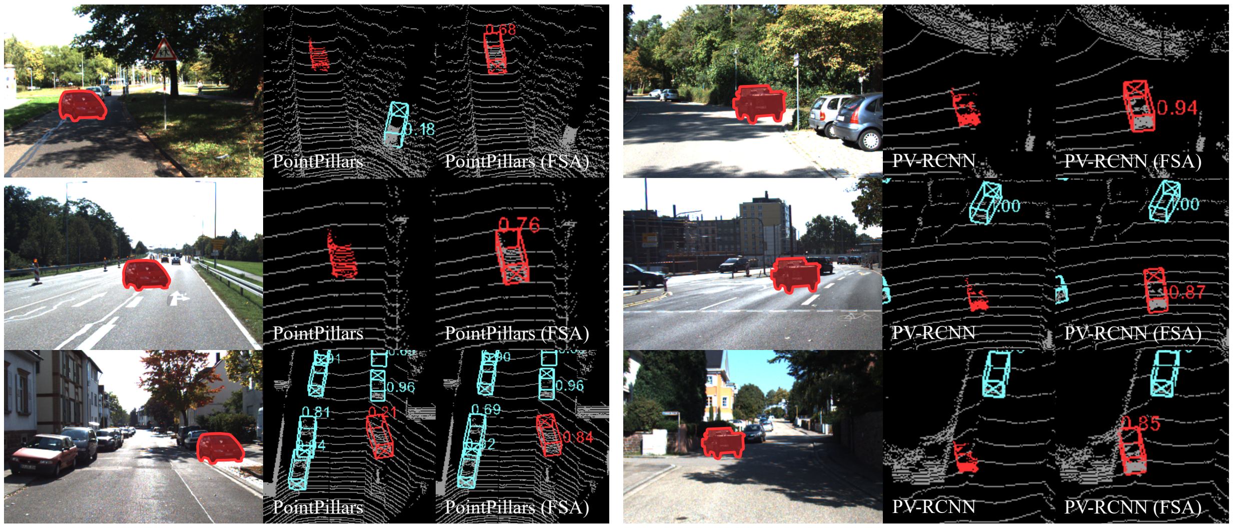


Figure 4.6: Performance illustrations on KITTI *val*. Red bounding box is ground truth, green is detector outputs. From left to right: (a) RGB images (b) Result of state-of-the-art methods: PointPillars [83], SECOND [184], Point-RCNN [144] and PV-RCNN [143]. (c) Result of our full self-attention (FSA) augmented baselines. Our method identifies missed detections and removes false positives.



(a) Object in the original scene



(b) Object inserted into another scene

Figure 4.7: (a) RGB images and point-clouds of cars on KITTI-*val* in which addition of context via FSA had the largest increase in the detection confidence. (b) We use a simple *copy and paste* method on these cars to create new point-clouds for testing our attention-based context aggregation detector for Point-Pillars and PV-RCNN backbone. We find that our FSA-based detector is more accurate and robust across scenes compared to the baseline.

Detailed Comparison with Baseline: In this section, we provide additional qualitative results across challenging scenarios from real-world driving scenes and compare them with the baseline performance (see Fig. 4.8). The ground-truth bounding boxes are shown in *red*, whereas the detector outputs are shown in *green*. We show consistent improvement in identifying missed detections across scenes and with different backbones including PointPillars [83], SECOND [184], Point-RCNN [144] and PV-RCNN [143]. We note that we can better refine proposal bounding box orientations with our context-aggregating FSA module (Rows 1, 2, and 4). We also note that cars at distant locations can be detected by our approach (Rows 3, 4 and 6). Finally we analyze that cars with slightly irregular shapes even at nearer distances are missed by the baseline but picked up by our approach (Rows 7 and 8).

4.5.5 Visualization of Attention Weights

We also visualize the attention weights for FSA-variant for the SECOND [184] backbone in Fig. 4.9. In this implementation, voxel features down-sampled by 8-times from the point-cloud space are used to aggregate context information through pairwise self-attention. We first visualize the voxel space, where the center point of each voxel is represented as a yellow point against the black scene-background. We next choose the center of a ground-truth bounding box as a reference point. We refer this bounding box as the reference bounding box. The reference bounding box is shown in *yellow*, and the rest of the labeled objects in the scene are shown in *orange*. We next visualize the attention weights across all the voxel centers with respect to the chosen reference bounding box center. Of the 4 attention maps produced by the 4 FSA-heads, we display the attention map with the largest activation in our figures. We find that attention weights become concentrated in small areas of the voxel-space. These voxel centers are called attended locations and are represented by a thick cross in our visualizations. The color of the cross represents the attention weight at that location and the scale of attention weights is represented using a colorbar. The size of the cross is manipulated manually by a constant factor. In an effort to improve image-readability, we connect the chosen reference object to the other labelled objects in the scene that it pays attention to (with blue boxes and blue arrows) as inferred from the corresponding attended locations while aggregating context information.

In our work, we speculate that sometimes for true-positive cases, CNNs (which are essentially a pattern matching mechanism) detect a part of the object but are not very confident about it. This confidence can be increased by looking at nearby voxels and inferring that the context-aggregated features resemble a composition of parts. We therefore first ask the question if our FSA module can adaptively focus on its own local neighbourhood. We show in Rows 1 and 2 of Fig. 4.9 that it can aggregate local context adaptively. We also hypothesize that, for distant cars, information from cars in similar lanes can help refine orientation. We therefore proceed to show instances where a reference bounding box can focus on cars in similar lanes, in Rows 3 and 4 of Fig. 4.9. We also show cases where FSA can adaptively focus on objects that are relevant to build structural information about the scene in Rows 5 and 6 of Fig. 4.9. Our visualizations thus indicate that semantically meaningful patterns emerge through the self-attention based context-aggregation module.

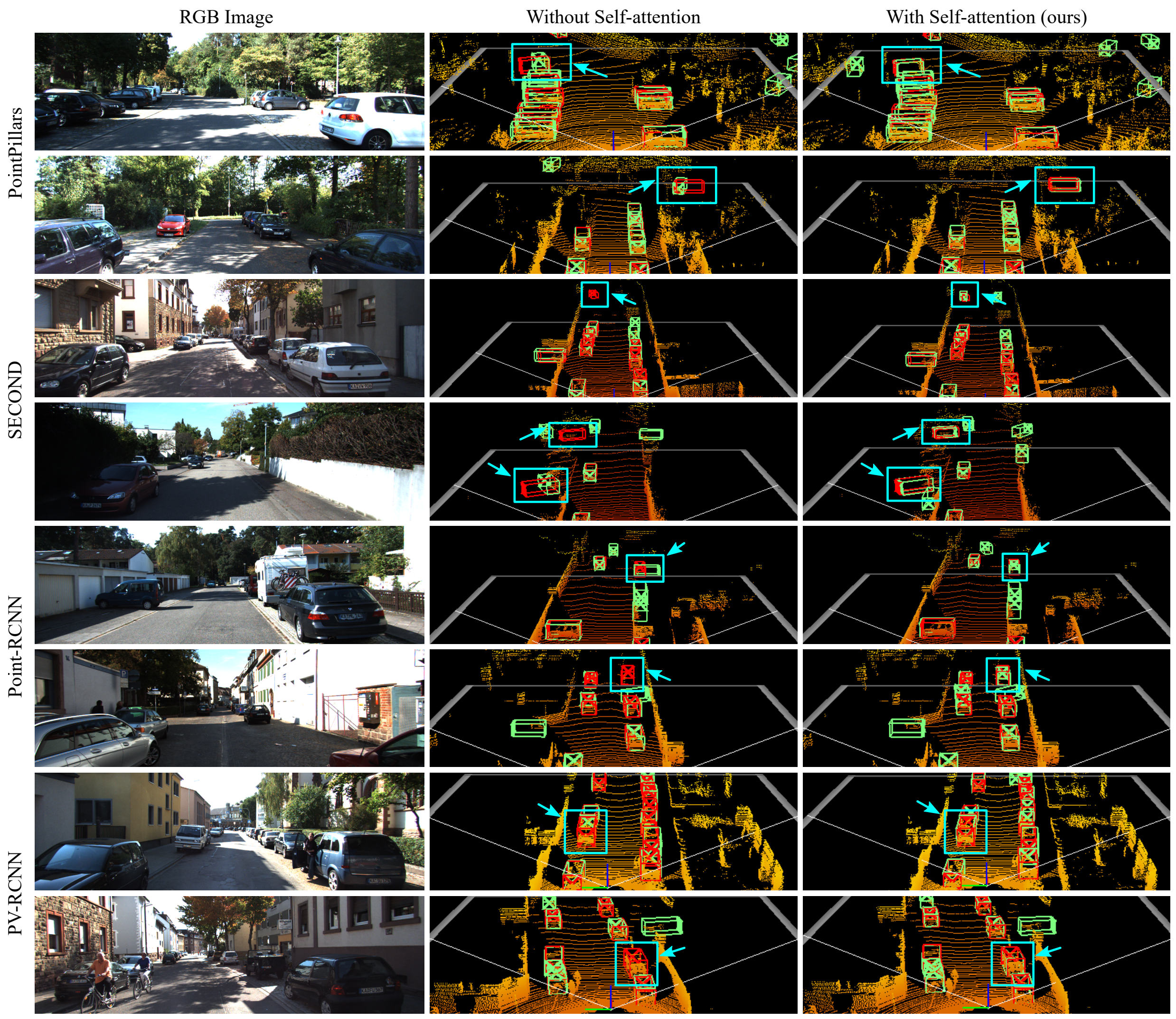


Figure 4.8: Qualitative comparisons of our proposed approach with the baseline on the KITTI validation set. *Red* represents Ground-Truth bounding box while *Green* represents detector outputs. From left to right: RGB images of scenes; Baseline performance across state-of-the-art detectors PointPillars [83], SECOND [184], Point-RCNN [144] and PV-RCNN [143]; Performance of proposed FSA module-augmented detectors. Viewed best when enlarged.

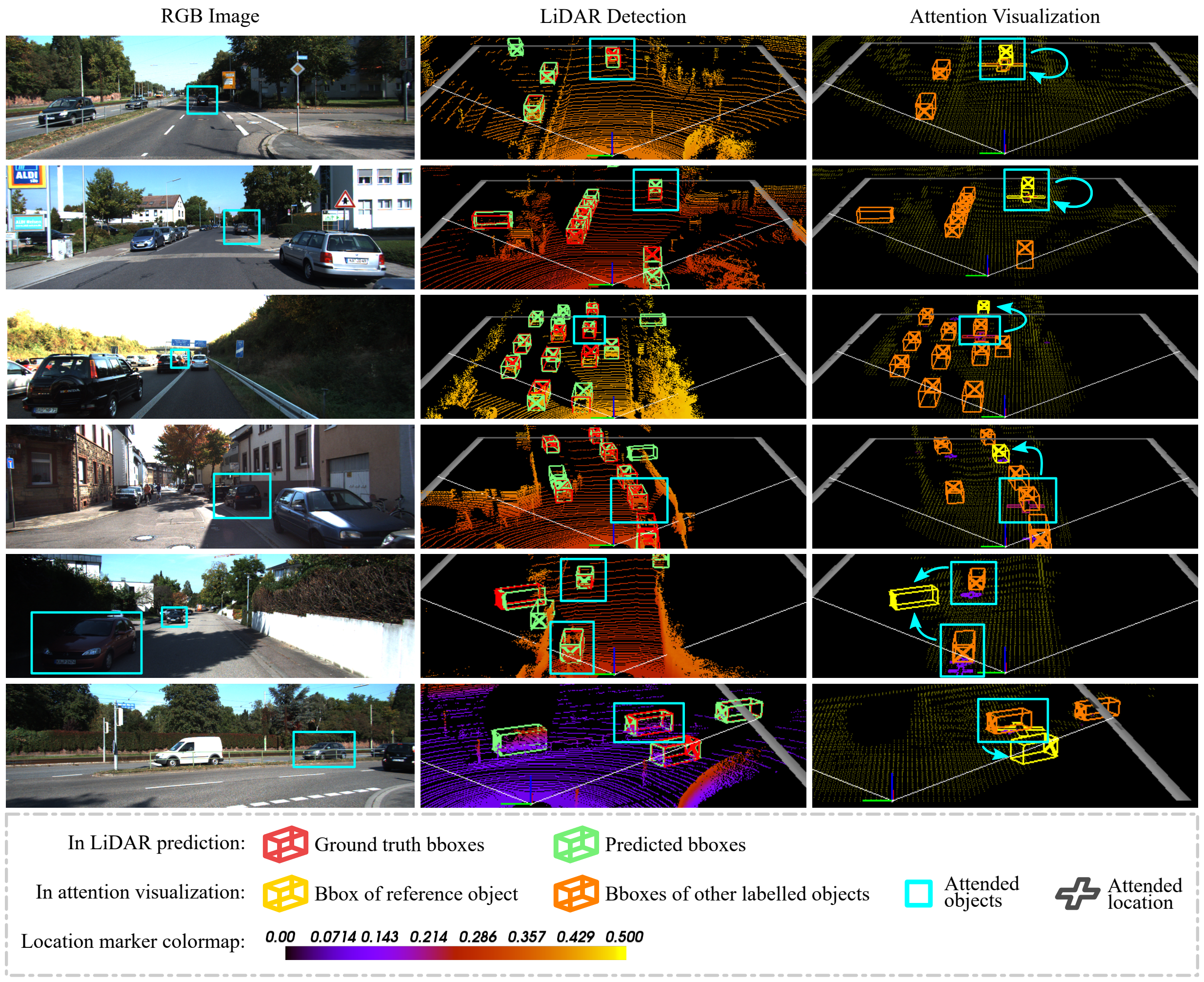


Figure 4.9: Visualization of attention maps produced by our proposed FSA-variant on SECOND [184] backbone. We analyze the implications of the produced attention maps in Sec. 4.5.5.

4.6 Chapter Conclusion

In this chapter, we propose a simple and flexible self-attention based framework to augment convolutional features with global contextual information for 3D object detection. Our proposed modules are generic, parameter and compute-efficient, and can be integrated into a range of 3D detectors. Our work explores two forms of self-attention: full (FSA) and deformable (DSA). The FSA module encodes pairwise relationships between all 3D entities, whereas the DSA operates on a representative subset to provide a scalable alternative for global context modeling. Quantitative and qualitative experiments demonstrate that our architecture systematically improves the performance of 3D object detectors.

In the next chapter, Chapter 5 we will explore how, given 3D object detections and object correspondences in the form of tracks, we can use it in an autonomous driving framework to forecast the future.

Part II

Predicting Trajectories of Surrounding Agents

Chapter 5

SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving

In the preceding chapters, Chapter 3 and Chapter 4, our primary focus was enhancing 3D object tracking and 3D object detection. However, the knowledge of surrounding objects is only part of the equation. Equally important is the ability to anticipate where these objects will be positioned in the future. Autonomous vehicles need to predict the future motion of other vehicles, pedestrians, and cyclists in the environment to plan their own path and maneuvers effectively and safely. Wrong predictions can result in false alarms, inefficient path planning, and in worst cases, accidents. Thus, it's crucial to develop accurate and robust motion forecasting models to avoid such potentially hazardous situations.

This chapter assumes that 3D object detections and object tracks are given, and focuses on improving motion forecasting methods thereby laying the groundwork for safer and more reliable autonomous driving systems. SSL-Lanes code and documentation is open-sourced.[1]

5.1 Motivation and Contributions

Motion forecasting in a real-world urban environment is an important task for autonomous robots. However, this is a very challenging problem. Difficulties include inherent stochas-

[1] `https://github.com/AutoVision-cloud/SSL-Lanes`

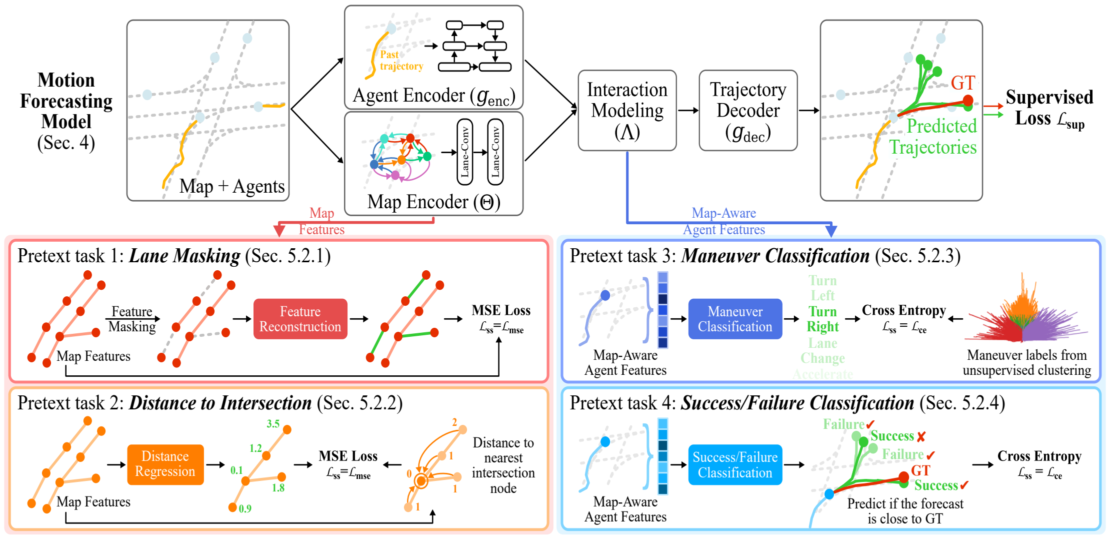
ticity and multimodality of driving behaviors, and that future motion can involve complicated maneuvers such as yielding, nudging, lane-changing, turning and acceleration or deceleration.

The motion prediction task has traditionally been based on kinematic constraints and road map information with handcrafted rules. These approaches however fail to capture long-term behavior and interactions with map structure and other traffic agents in complex scenarios. Tremendous progress has been made with data-driven methods in motion forecasting [25, 105, 205, 51, 73, 111, 98, 24]. Recent methods use a vector representation for HD maps and agent trajectories, including approaches like Lane-GCN [88], Lane-RCNN [197], Vector-Net [41], TNT [205] and Dense-TNT [51]. More recently, the enormous success of transformers [164] has been leveraged for forecasting in mm-Transformer [98], Scene transformer [111], Multimodal transformer [66] and Latent Variable Sequential Transformers [47]. Most of these methods however are extremely complex in terms of architecture and have low inference speeds, which makes them unsuitable for real-world settings.

In this chapter, we extend ideas from self-supervised learning (SSL) to the motion forecasting task. Self-supervision has seen huge interest in both natural language processing and computer vision [21] to make use of freely available data without the need for annotations. It aims to assist the model to learn more transferable and generalized representation from pseudo-labels via pretext tasks. Given the recent success of self-supervision with CNNs, transformers, and GNNs, we are naturally motivated to ask the question: *Can self-supervised learning improve accuracy and generalizability of motion forecasting, without sacrificing inference speed or architectural simplicity?*

Contributions: Our work, SSL-Lanes, presents the first systematic study on how to incorporate self-supervision in a standard data-driven motion forecasting model. Our contributions are:

- We demonstrate the effectiveness of incorporating self-supervised learning in motion forecasting. Since this does not add extra parameters or compute during inference, SSL-Lanes achieves the best accuracy-simplicity-efficiency trade-off on the challenging large-scale Argoverse [26] benchmark.
- We propose four self-supervised tasks based on the nature of the motion forecasting problem. The key idea is to leverage easily accessible map/agent-level information to define domain-specific pretext tasks that encourage the standard model to capture more superior and generalizable representations for forecasting, in comparison to pure supervised learning.



Motion Forecasting Model (Sec. 4)
Map + Agents
Past trajectory
Agent Encoder (g_{enc})
Lane-Conv
Lane-Conv
Map Encoder (Θ)
Interaction Modeling (Λ)
Trajectory Decoder (g_{dec})
GT
Predicted Trajectories
Supervised Loss $\mathcal{L}_{\text{sup}}$
Map Features
Map-Aware Agent Features
Pretext task 1: Lane Masking (Sec. 5.2.1)
Feature Masking
Feature Reconstruction
MSE Loss $\mathcal{L}_{ss}=\mathcal{L}_{mse}$
Map Features
Pretext task 2: Distance to Intersection (Sec. 5.2.2)
Distance Regression
MSE Loss $\mathcal{L}_{ss}=\mathcal{L}_{mse}$
Distance to nearest intersection node
Map Features
Pretext task 3: Maneuver Classification (Sec. 5.2.3)
Map-Aware Agent Features
Maneuver Classification
Turn Left
Turn Right
Lane Change
Accelerate
Cross Entropy $\mathcal{L}_{ss}=\mathcal{L}_{ce}$
Maneuver labels from unsupervised clustering
Pretext task 4: Success/Failure Classification (Sec. 5.2.4)
Map-Aware Agent Features
Success/Failure Classification
Failure ✓
Success ✗
Failure ✓
GT
Success ✓
Predict if the forecast is close to GT
Cross Entropy $\mathcal{L}_{ss}=\mathcal{L}_{ce}$

Figure 5.1: Illustration of the overall SSL-Lanes framework for self-supervision on motion forecasting through joint training. SSL-Lanes improves upon a standard-motion forecasting baseline, that consists of an agent encoder, map encoder, interaction model and a trajectory decoder, trained using a supervised loss $\mathcal{L}_{sup}$. SSL-Lanes proposes four pretext tasks: (1) Lane Masking: which recovers feature information from the perturbed lane graphs. (2) Distance to Intersection: which predicts the distance (in terms of shortest path length) from all lane nodes to intersection nodes. (3) Maneuver Classification: predicts the form of a 'maneuver' the agent-of-interest intends to execute (4) Success/Failure Classification: which trains an agent specialized at achieving end-point goals.

- We further design experiments to explore why forecasting benefits from SSL. We provide extensive results to hypothesize that SSL-Lanes learns richer features from the SSL training as compared to a model trained with vanilla supervised learning.

5.2 Problem Formulation

We are given the past motion of N actors. The i-th actor is denoted as a set of its center locations over the past L time-steps. We pre-process it to represent each trajectory as a sequence of displacements $\mathcal{P}_i = \{\Delta \boldsymbol{p}_i^{-L+1}, ..., \Delta \boldsymbol{p}_i^{-1}, \Delta \boldsymbol{p}_i^{0}\}$, where $\boldsymbol{p}_i^l$ is the 2D displacement from time step $l-1$ to l. We are also given a high-definition (HD) map, which contains lanes and semantic attributes. Each lane is composed of several consecutive lane nodes, with a total of M nodes. $\boldsymbol{X} \in \mathbb{R}^{M\times F}$ denotes the lane node feature matrix, where $\boldsymbol{x}_j = \boldsymbol{X}[j,:]^T$ is the F-dimensional lane node vector. Following the connections between lane centerlines (i.e., predecessor, successor, left neighbour and right neighbour), we represent the connectivity among the lane nodes with four adjacency matrices $\{\boldsymbol{A}_f\}_{f\in\{\text{pre,suc,left,right}\}}$, with $\boldsymbol{A}_f \in \mathbb{R}^{M\times M}$. This implies that if $\boldsymbol{A}_{f,gh} = 1$, then node h is an f-type neighbor of node g. Our goal is to forecast the future motions of all actors in the scene $\mathcal{O}^{1:T}_{\text{GT}} = \{(x_i^1, y_i^1), ..., (x_i^T, y_i^T)|i = 1, ..., N\}$, where T is our prediction horizon.

5.3 Background

We first briefly introduce a standard data-driven motion forecasting framework.

Feature Encoding: We first encode the agent and map inputs similar to Lane-GCN [88]. The agent encoder includes a 1D convolution with a feature pyramid network, parameterized by g_{enc}, as given by Eq. (5.1). For map-encoding, we adopt two Lane-Conv residual blocks, parameterized by $\boldsymbol{\Theta} = \{\boldsymbol{W}_0, \boldsymbol{W}_{\text{left}}, \boldsymbol{W}_{\text{right}}, \boldsymbol{W}_{\text{pre},k}, \boldsymbol{W}_{\text{suc},k}\}$, where $k \in \{1, 2, 4, 8, 16, 32\}$, as given by Eq. (5.2).

$$\hat{\boldsymbol{p}}_i = g_{\text{enc}}(\mathcal{P}_i) \tag{5.1}$$

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}_0 + \sum_{j\in\{\text{left,right}\}} \boldsymbol{A}_j\boldsymbol{X}\boldsymbol{W}_j + \sum_k \boldsymbol{A}^k_{\text{pre}}\boldsymbol{X}\boldsymbol{W}_{\text{pre},k} + \boldsymbol{A}^k_{\text{suc}}\boldsymbol{X}\boldsymbol{W}_{\text{suc},k} \tag{5.2}$$

Modeling Interactions: Since the behavior of agents depends on map topology and social consistency, each encoded agent i subsequently aggregates context from the surrounding map features and its neighboring agent features, via spatial attention [164] as given by Eq. (5.3):

$$\begin{aligned} \tilde{\boldsymbol{p}}_i &= \hat{\boldsymbol{p}}_i\boldsymbol{W}_{\text{M2A}} + \sum_j \phi(\text{concat}(\hat{\boldsymbol{p}}_i, \Delta_{i,j}, \boldsymbol{y}_j)\boldsymbol{W}_1)\boldsymbol{W}_2 \\ \acute{\boldsymbol{p}}_i &= \tilde{\boldsymbol{p}}_i\boldsymbol{W}_{\text{A2A}} + \sum_j \phi(\text{concat}(\tilde{\boldsymbol{p}}_i, \Delta_{i,j}, \tilde{\boldsymbol{p}}_j)\boldsymbol{W}_3)\boldsymbol{W}_4 \end{aligned} \tag{5.3}$$

Here, $\boldsymbol{y}_j$ is the feature of the j-th node, $\hat{\boldsymbol{p}}_i$ is the feature of the i-th agent, ϕ the composition of layer normalization and ReLU, and $\Delta_{ij} = \text{MLP}(\boldsymbol{v}_j - \boldsymbol{v}_i)$, where $\boldsymbol{v}$ denotes the (x, y) 2-D BEV location of the agent or the lane node. The parameters for map and agent feature aggregation is represented by $\boldsymbol{\Lambda} = \{\boldsymbol{W}_{\text{M2A}}, \boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_{\text{A2A}}, \boldsymbol{W}_3, \boldsymbol{W}_4\}$.

Trajectory Prediction: Finally, we decode the future trajectories from the features $\acute{\boldsymbol{p}}_i$ corresponding to the agents of interest as given by: $\mathcal{O}^{1:T}_{\text{pred}} = \{g_{\text{dec}}(\acute{\boldsymbol{p}}_i)|i = 1, ..., N\}$, where

SSL Task	**Property Level**	**Primary Assumption**	**Type**
Lane-Masking	Map features	Local map structure	Aux. auto-encoder
Distance to Intersection		Global map structure	Aux. regression
Maneuver Classification	Map-aware agent features	Agent feature similarity	Aux. classification
Success/Failure Classification		Distance to success state	

Table 5.1: Overview of our proposed self-supervised (SSL) tasks

g_{dec} is the parameterized trajectory decoder. The parameters for the motion forecasting model are learned by minimizing the supervised loss ($\mathcal{L}_{\text{sup}}$) calculated between the predicted output and the ground-truth future trajectories ($\mathcal{O}_{\text{GT}}^{1:T}$), as given by Eq. (5.4):

$$g_{\text{enc}}^{\star}, \mathbf{\Theta}^{\star}, \mathbf{\Lambda}^{\star}, g_{\text{dec}}^{\star} = \underset{g_{\text{enc}}, \mathbf{\Theta}, \mathbf{\Lambda}, g_{\text{dec}}}{\arg\min} \ \mathcal{L}_{\text{sup}}(\mathcal{O}_{\text{pred}}^{1:T}, \mathcal{O}_{\text{GT}}^{1:T}) \tag{5.4}$$

5.4 SSL-Lanes

The goal of our proposed SSL-Lanes framework is to improve the performance of the primary motion forecasting baseline by learning simultaneously with various self-supervised tasks. Fig. 5.1 shows the pipeline of our proposed approach, and Tab. 5.1 summarizes the self-supervised tasks.

5.4.1 Self-Supervision meets Motion Forecasting

Before we discuss designing pretext tasks to generate self-supervisory signals, we consider a scheme that will allow combined training for self-supervised pretext tasks and our standard framework.

How to combine motion forecasting and SSL? Self-supervision can be combined with motion forecasting in various ways. In one scheme we could pre-train the forecasting encoder with pretext tasks (which can be viewed as an initialization for the encoder's parameters) and then fine-tune the pre-trained encoder with a downstream decoder as given by Eq. (5.4). In another scheme, we could choose to freeze the encoder and only train the decoder. In a third scheme, we could optimize our pretext task and primary task *jointly*, as a kind of multi-task learning setup. Inspired by relevant discussions in GNNs, we choose the third-scheme, i.e., multi-task learning, which is the most general framework among the three and is also experimentally verified to be the most effective [193, 70].

Joint Training: Considering our motion forecasting task and a self-supervised task, the output and the training process can be formulated as:

$$\mathbf{\Psi}^\star, \mathbf{\Omega}^\star, \mathbf{\Theta}_{\text{ss}}^\star = \underset{\mathbf{\Psi}, \mathbf{\Omega}, \mathbf{\Theta}_{\text{ss}}}{\arg\min} \quad \alpha_1 \mathcal{L}_{\text{sup}}(\mathbf{\Psi}, \mathbf{\Omega}) + \alpha_2 \mathcal{L}_{\text{ss}}(\mathbf{\Psi}, \mathbf{\Theta}_{\text{ss}}) \tag{5.5}$$

where, $\mathcal{L}_{\text{ss}}(\cdot, \cdot)$ is the loss function of the self-supervised task, $\mathbf{\Theta}_{\text{ss}}$ is the corresponding linear transformation parameter, and $\alpha_1, \alpha_2 \in \mathrm{R}_{>0}$ are the weights for the supervised and self-supervised losses. If the pretext task only focuses on the map encoder, then $\mathbf{\Psi} = \{\mathbf{\Theta}\}$ and $\mathbf{\Omega} = \{g_{\text{enc}}, \mathbf{\Lambda}, g_{\text{dec}}\}$. Otherwise, $\mathbf{\Psi} = \{g_{\text{enc}}, \mathbf{\Theta}, \mathbf{\Lambda}\}$ and $\mathbf{\Omega} = \{g_{\text{dec}}\}$. Henceforth, we also define the following representations. We will represent the primary task encoder as function $f_{\mathbf{\Psi}}$, parameterized by $\mathbf{\Psi}$. Furthermore, given a pretext task, which we will design in the next section, the pretext decoder $p_{\mathbf{\Theta}_{\text{ss}}}$ is a function that predicts pseudo-labels and is parameterized by $\mathbf{\Theta}_{\text{ss}}$.

Benefit of SSL-Lanes: In Eq. (5.5), the self-supervised task as a regularization term throughout network training. It acts as the regularizer learned from unlabeled data under the minor guidance of human prior (design of pretext task). Therefore, a properly designed task would introduce data-driven prior knowledge that improves model generalizability.

5.4.2 Pretext tasks for Motion Forecasting

At the core of our SSL-Lanes approach is defining pretext tasks based upon self-supervised information from the underlying map structure *and* the overall temporal prediction problem itself. Our proposed prediction-specific self-supervised tasks are summarized in Tab. 5.1, and assign different pseudo-labels from unannotated data to solve Eq. (5.5). Our core approach is simple in contrast to state-of-the-art that rely on complex encoding architectures [197, 98, 111, 73, 88, 66, 205], ensembling forecasting heads [190, 163], involved final goal-set optimization algorithms [51, 152] or heavy fusion mechanisms [88], to improve prediction performance.

A. Lane-Masking

Motivation: The goal of the *Lane-Masking* pretext task is to encourage the map encoder $\mathbf{\Psi} = \{\mathbf{\Theta}\}$ to learn local structure information in addition to the forecasting task that is being optimized. In this task, we learn by recovering feature information from the perturbed lane graphs. VectorNet [41] is the only other motion forecasting work that proposes to randomly mask out the input node features belonging to either scene context or agent trajectories, and ask the model to reconstruct the masked features. Their intuition is to encourage the graph networks to better capture the interactions between agent dynamics and scene context. However, our motivation differs from VectorNet in two respects: (a) We propose to use masking to learn local map-structure better, as opposed to learning interactions between map and the agent. This is an easier optimization task, and we outperform VectorNet. (b) A lane is made up of several nodes. We propose to randomly mask out a certain percentage of each lane. This is a much stronger prior as compared to randomly masking out *any* node and ensures that the model pays attention to all parts of the map.

Formulation: Formally, we randomly mask (i.e., set equal to zero) the features of m_a percent of nodes per lane and then ask the self-supervised decoder to reconstruct these features.

$$\boldsymbol{\Psi}^{\star}, \boldsymbol{\Theta}_{\text{ss}}^{\star} = \underset{\boldsymbol{\Psi}, \boldsymbol{\Theta}_{\text{ss}}}{\arg\min} \frac{1}{m_a} \sum_{i=1}^{m_a} \mathcal{L}_{\text{mse}} \Big(p_{\boldsymbol{\Theta}_{\text{ss}}}([f_{\boldsymbol{\Psi}}(\tilde{\boldsymbol{X}}, \boldsymbol{A}_f)]_{\boldsymbol{v}_i}), \boldsymbol{X}_i \Big) \tag{5.6}$$

Here, $\tilde{\boldsymbol{X}}$ is the node feature matrix corrupted with random masking, i.e., some rows of $\boldsymbol{X}$ corresponding to nodes $\boldsymbol{v}_i$ are set to zero. $p_{\boldsymbol{\Theta}_{\text{ss}}}$ is a fully connected network that maps the representations to the reconstructed features. $\mathcal{L}_{\text{mse}}$ is the mean squared error (MSE) loss function penalizing the distance between the reconstructed map features $p_{\boldsymbol{\Theta}_{\text{ss}}}([f_{\boldsymbol{\Psi}}(\tilde{\boldsymbol{X}}, \boldsymbol{A}_f)]_{\boldsymbol{v}_i})$ for node $\boldsymbol{v}_i$ and its GT features $\boldsymbol{X}_i$.

Benefit of Lane-Masking: Since Argoverse [26] has imbalanced data with respect to maneuvers, there are cases when right/left turns, lane-changes, acceleration/deceleration are missed by the baseline even with multi-modal predictions. We hypothesize that stronger map-features can help the multi-modal prediction header to infer that some of the predictions should also be aligned with map topology. For example, even if an agent is likely to go straight at an intersection, some of the possible futures should also cover acceleration/deceleration or right/left turns guided by the local map structure.

B. Distance to Intersection

Motivation: The Lane-Masking pretext task is from a local structure perspective based on masking and trying to predict local attributes of the vectorized HD-map. We further develop the *Distance-to-Intersection* pretext task to guide the map-encoder, $\boldsymbol{\Psi} = \{\boldsymbol{\Theta}\}$, to maintain global topology information by predicting the distance (in terms of shortest path length) from all lane nodes to intersection nodes. Datasets like Argoverse [26] provide lane attributes which describe whether a lane node is located within an intersection. This will force the representations to learn a global positioning vector of each of the lane nodes.

Formulation: We aim to regress the distances from each lane node to pre-labeled intersection nodes annotated as part of the dataset. Given K labeled intersection nodes $\mathcal{V}_{\text{intersection}} = \{\boldsymbol{v}_{\text{intersection},k} | k = 1, ...K\}$, we first generate reliable pseudo labels using breadth-first search (BFS). Specifically, BFS calculates the shortest distance $d_i \in \mathbb{R}$ for every lane node $\boldsymbol{v}_i$ from the given set $\mathcal{V}_{\text{intersection}}$. The target of this task is to predict the pseudo-labeled distances using a pretext decoder. If $p_{\boldsymbol{\Theta}_{\text{ss}}}([f_{\boldsymbol{\Psi}}(\boldsymbol{X}, \boldsymbol{A}_f)]_{\boldsymbol{v}_i})$ is the prediction of node $\boldsymbol{v}_i$, and $\mathcal{L}_{\text{mse}}$ is the mean-squared error loss function for regression, then the loss formulation for this SSL pretext task is as follows:

$$\boldsymbol{\Psi}^\star, \boldsymbol{\Theta}_{\text{ss}}^\star = \arg\min_{\boldsymbol{\Psi}, \boldsymbol{\Theta}_{\text{ss}}} \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}_{\text{mse}}\Big(p_{\boldsymbol{\Theta}_{\text{ss}}}([f_{\boldsymbol{\Psi}}(\boldsymbol{X}, \boldsymbol{A}_f)]_{\boldsymbol{v}_i}), d_i\Big) \tag{5.7}$$

Benefit of Distance to Intersection Task: We hypothesize that since change of speed, acceleration, primary direction of movement etc. for an agent can change far more dramatically as an agent approaches or moves away from an intersection, it is beneficial to explicitly incentivize the model to pick up the geometric structure near an intersection and compress the space of possible map-feature encoders, thereby effectively simplifying inference. We also expect this to improve drivable area compliance nearby an intersection, which is often a problem for current motion forecasting models.

C. Maneuver Classification

Motivation: The Lane-Masking and Distance to Intersection pretext tasks are both based on extracting feature and topology information from a HD-map. However, pretext tasks can also be constructed from the overall forecasting task itself. Thus we propose to obtain free pseudo-labels in the form of a 'maneuver' the agent-of-interest intends to execute, and define a set of 'intentions' to represent common semantic modes (e.g. change lane, speed up, slow down, turn-right, turn-left etc.) We call this pretext task *Maneuver Classification*, and we expect it to provide prior regularization to $\boldsymbol{\Psi} = \{g_{\text{enc}}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}\}$, based on driving modes.

Formulation: We aim to construct pseudo label to divide agents into different clusters according to their driving behavior and explore unsupervised clustering algorithms to acquire the maneuver for each agent. We find that using naive k-Means (on agent end-points) or DBSCAN (on Hausdorff distance between entire trajectories [2]) leads to noisy clustering. We find that constrained k-means [166] on agent end-points works best to divide trajectory samples into C clusters equally. We define $C = \{\text{maintain-speed}, \text{accelerate}, \text{decelerate}, \text{turn-left}, \text{turn-right},$

lane-change} and the clustering function as ρ. If $p_{\boldsymbol{\Theta}_{\text{ss}}}(f_{\boldsymbol{\Psi}}(\mathcal{P}_i, \boldsymbol{X}, \boldsymbol{A}_f))$ is the prediction of agent i's intention and $E_i = (x^T_{i,\text{GT}}, y^T_{i,\text{GT}})$ is its ground-truth end-point, then the learning objective is to classify each agent maneuver into its corresponding cluster using cross-entropy loss $\mathcal{L}_{ce}$ as:

$$\boldsymbol{\Psi}^\star, \boldsymbol{\Theta}^\star_{\text{ss}} = \underset{\boldsymbol{\Psi}, \boldsymbol{\Theta}_{\text{ss}}}{\arg\min}\, \mathcal{L}_{\text{ce}}\Big(p_{\boldsymbol{\Theta}_{\text{ss}}}(f_{\boldsymbol{\Psi}}(\mathcal{P}_i, \boldsymbol{X}, \boldsymbol{A}_f)), \rho(E_i)\Big) \tag{5.8}$$

Benefit of Maneuver Classification Task: We hypothesize if one can identify the intention of a driver, the future motion of the vehicle will match that maneuver, thereby reducing the set of possible end-points for the agent. We also expect that agents with similar maneuvers will tend to have consistent semantic representations.

D. Forecasting Success/Failure Classification

Motivation: In contrast to maneuver classification, which provides coarse-grained prediction of the future, self-supervision mechanisms can also offer a strong learning signal through goal-reaching tasks which are generated from the agent's trajectories. We propose a pretext task called *Success/Failure Classification*, which trains an agent specialized at achieving end-point goals which directly lead to the forecasting-task solution. We expect this to constrain $\boldsymbol{\Psi} = \{g_{\text{enc}}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}\}$ to predict trajectories ϵ distance away from the correct final end-point. Conceptually, the more examples of successful goal states we collect, the better understanding of the target goal of the forecasting task we have.

Formulation: Similar to maneuver classification, we wish to create pseudo-labels for our data samples. We label trajectory predictions as successful ($c = 1$) if the final prediction $(x^T_{i,\text{pred}}, y^T_{i,\text{pred}})$ is within $\epsilon < 2$m of the final end-point E_i, and as failure ($c = 0$) otherwise. We choose 2m as our ϵ threshold because it is also used for miss-rate calculation (Sec. 5.5). In this case, $c \in C = \{0, 1\}$ is the pseudo-label which belongs to label set C. If the pretext decoder predicts agent i's final-endpoint as $p_{\boldsymbol{\Theta}_{\text{ss}}}(f_{\boldsymbol{\Psi}}(\mathcal{P}_i, \boldsymbol{X}, \boldsymbol{A}_f))$, and given ground-truth end-point E_i its success or failure label is c_i, then the pretext loss can be formulated as:

$$\boldsymbol{\Psi}^\star, \boldsymbol{\Theta}^\star_{\text{ss}} = \underset{\boldsymbol{\Psi}, \boldsymbol{\Theta}_{\text{ss}}}{\arg\min}\, \mathcal{L}_{\text{ce}}\Big(p_{\boldsymbol{\Theta}_{\text{ss}}}(f_{\boldsymbol{\Psi}}(\mathcal{P}_i, \boldsymbol{X}, \boldsymbol{A}_f)), c_i\Big) \tag{5.9}$$

Benefit of Success/Failure Classification Task: We hypothesize that this task will especially provide stronger gains for cases where the final end-point is not aligned with the general direction of agent movement for majority of samples given in the dataset, and is thus not well captured by average displacement based supervised loss functions.

5.4.3 Learning

As all the modules are differentiable, we can train the model in an end-to-end way. We use the sum of classification, regression and self-supervised losses to train the model. Specifically, we use:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{terminal}} + \mathcal{L}_{\text{ss}} \tag{5.10}$$

For classification and regression loss design, we adopt the formulation proposed in [88]. $\mathcal{L}_{\text{terminal}} = \frac{1}{N}\sum_{i=1}^{N} L2\Big((x^T_{i,\text{pred}}, y^T_{i,\text{pred}}), (x^T_{i,\text{GT}}, y^T_{i,\text{GT}})\Big)$ is a simple L2 loss that minimizes the distance between predicted final-endpoints and the ground-truth. This is because $\mathcal{L}_{\text{reg}}$ is averaged across all time-points $1:T$, and from a practical end user perspective, minimizing the endpoint loss is much more important than weighting loss from all time-steps equally. Our proposed pretext tasks contributes to $\mathcal{L}_{\text{ss}}$. During evaluation, we study each pretext task separately, and their corresponding loss formulations defined in Eq. (5.6), Eq. (5.7), Eq. (5.8), Eq. (5.9) are used for joint training.

5.5 Experiments

We present the experimental and implementation details in this section.

Dataset: Argoverse provides a large-scale dataset [26] for the purpose of training, validating and testing models, where the task is to forecast 3 seconds of future motions, given 2 seconds of past observations. This dataset has more than 30K real-world driving sequences collected in Miami (MIA) and Pittsburgh (PIT). Those sequences are further split into train, validation, and test sets, without any geographical overlap. Each of them has 205,942, 39,472, and 78,143 sequences respectively. In particular, each sequence contains the positions of all actors in a scene within the past 2 seconds history, annotated at 10Hz. It also specifies one actor of interest in the scene, with type 'agent', whose future 3 seconds of motion are used for the evaluation. The train and validation splits additionally provide future locations of all actors within 3 second horizon labeled at 10Hz, while annotations for test sequences are withheld from the public and used for the leaderboard evaluation. HD map information is available for all sequences.

We have two main requirements for the dataset: (a) **Scale of Data:** Modern motion forecasting methods and self-supervised learning systems require a large amount of training data to imitate human maneuvers in complex real-world scenarios. Thus, the dataset should be *large-scale* and *diverse*, such that it has a wide range of behaviors and trajectory shapes across different geometries represented in the data. (b) **Interesting Scenarios**

for Forecasting Evaluation: The dataset should be collected for interesting behaviours by biasing sampling towards complex observed behaviours (e.g., lane changes, turns) and road features (e.g., intersections), since we wish to focus on these cases. We find that on the basis of these requirements, as well as its popularity in the the motion forecasting community, Argoverse [26] is the best candidate to showcase our method. Please refer to the supplementary for more details regarding why we choose to focus on it in comparison to other motion forecasting benchmarks.

Metrics: ADE is defined as the average displacement error between ground-truth trajectories and predicted trajectories over all time steps. FDE is defined as displacement error between ground-truth trajectories and predicted trajectories at the final time step. We compute K likely trajectories for each scenario with the ground truth label, where $K = 1$ and $K = 6$ are used. Therefore, minADE and minFDE are minimum ADE and FDE over the top K predictions, respectively. Miss rate (MR) is defined as the percentage of the best-predicted trajectories whose FDE is within a threshold (2 m). Brier-minFDE is the minFDE plus $(1 - p)^2$, where p is the corresponding trajectory probability.

Experimental Details: To normalize the data, we translate and rotate the coordinate system of each sequence so that the origin is at current position $t = 0$ of 'agent' actor and x-axis is aligned with its current direction, i.e., orientation from the agent location at $t = -1$ to the agent location at $t = 0$ is the positive x axis. We use all actors and lanes whose distance from the agent is smaller than 100 meters as the input. We train the model on 4 TITAN-X GPUs using a batch size of 128 with the Adam [75] optimizer with an initial learning rate of 1×10^{-3}, which is decayed to 1×10^{-4} at 100,000 steps. The training process finishes at 128,000 steps and takes about 10 hours to complete. For our final test-set submission, we use success/failure classification as the pretext task, and initialize the map-encoder with the parameters from a model trained with the lane-masking pretext task. To avoid overfitting to the general directions that agents move, we augment the data from each scene for the test-set submission. We rotate all trajectories in a scene around the scene's origin by γ, where γ varies from $0°$ to $360°$ in $30°$ intervals. We provide more implementation details in the supplementary.

5.6 Results

5.6.1 Ablation Studies

Effectiveness of Pretext tasks: We first examine the effect of incorporating our proposed pretext tasks (Sec. 5.4) with the standard data-driven motion forecasting baseline

Method	minADE$_1$	minFDE$_1$	MR$_1$	minADE$_6$	minFDE$_6$	MR$_6$
Baseline	1.42	3.18	51.35	0.73	1.12	11.07
Lane-Masking	1.36	2.96	49.45	**0.70**	1.02	8.82
Distance to Intersection	1.38	3.02	49.53	0.71	1.04	8.93
Maneuver Classification	**1.33**	**2.90**	49.26	0.72	1.05	9.36
Success/Failure Classification	1.35	2.93	**48.54**	**0.70**	**1.01**	**8.59**

Table 5.2: Motion forecasting performance on Argoverse validation with our proposed pretext tasks

(Sec. 5.3). While evaluating the importance of our proposed pretext tasks, we wish to underline that motion prediction for autonomous driving is a safety-critical task, especially at intersections where most of our data is collected, and most accidents also happen. We thus posit that in this situation, even a small error in predicting final locations (FDE) for a given agent can lead to dangerous potential collisions.

Results in Tab. 5.2 show that all proposed pretext tasks improve motion forecasting performance for Argoverse. Specifically, the lane-masking pretext task improves min-FDE by 8.9% and MR@2m by 20.3%. distance to intersection improves min-FDE by 7.1% and 19.3%. Maneuver classification improves min-FDE by 6.3% and MR@2m by 15.4%. We expect that improving the quality of clustering for maneuvers and thus creating better pseudo-labels will improve this further. Finally, success/failure classification improves min-FDE by 9.8% and MR@2m by 22.4%. Moreover, since pretext tasks are not used for inference and only for training, they also do not add any extra parameters or FLOPs to the baseline, thereby increasing accuracy but at no cost to computational efficiency or architectural complexity. We present qualitative results with the different pretext tasks on several hard cases in Fig. 5.4.

Similarity in feature space: We analyze the CKA similarity [77] between the representations learnt by: a model trained with pretext task 'D2I' (refers to distance to intersection task) and baseline; two models trained with different pretext tasks. In Fig. 5.3, Base(M2A) refers to $\tilde{\boldsymbol{p}}_i$, Base(A2A) refers to $\acute{\boldsymbol{p}}_i$ (see Eq. (5.3)), 'Mask' refers to lane-masking, 'success/fail' refers to success or failure classification task and 'intention' suggests maneuver classification.

Our main questions are: (a) how much does the pretext task feature differ from the baseline? (b) do the features from different pretext tasks collapse to the same feature? First we note that representation learned by D2I does not *collapse* to the same representation learned by Mask or Success/Fail or Intention. Secondly we note that D2I features are

Method	**minADE$_1$**	**minFDE$_1$**	**MR$_1$**	**minADE$_6$**	**minFDE$_6$**	**MR$_6$**	**b-FDE$_6$**
NN + Map [26]	3.65	8.12	94.0	2.08	4.02	58.0	-
Jean [105]	1.74	4.24	68.56	0.98	1.42	13.08	2.12
Lane-GCN [88]	1.71	3.78	58.77	0.87	1.36	16.20	2.05
LaneRCNN [197]	1.68	3.69	56.85	0.90	1.45	12.32	2.15
TNT [205]	1.77	3.91	59.70	0.94	1.54	13.30	2.14
DenseTNT [51]	1.68	3.63	58.43	0.88	1.28	12.58	1.97
PRIME [152]	1.91	3.82	58.67	1.22	1.55	11.50	2.09
WIMP [73]	1.82	4.03	62.88	0.90	1.42	16.69	2.11
TPCN [189]	1.66	3.69	58.80	0.87	1.38	15.80	1.92
HOME [45]	1.70	3.68	57.23	0.89	1.29	**8.46**	**1.86**
mmTransformer [98]	1.77	4.00	61.78	0.87	1.34	15.40	2.03
MultiModalTransformer [66]	1.74	3.90	60.23	0.84	1.29	14.29	1.94
LatentVariableTransformer [47]	-	-	-	0.89	1.41	16.00	-
SceneTransformer [111]	1.81	4.06	59.21	**0.80**	**1.23**	12.55	1.88
SSL-Lanes (Ours)	**1.63**	**3.56**	**56.71**	0.84	1.25	13.26	1.94

Table 5.3: Comparison of our (best) proposed model and top approaches on the Argoverse Test. The best results are in bold and underlined, and the second best is also underlined.

quite different from Base-M2A features $\tilde{\boldsymbol{p}}_i$ and Base-A2A features $\acute{\boldsymbol{p}}_i$, which suggests that *task-specific regularization* has indeed resulted in different parameters.

5.6.2 Comparison with State-of-the-Art

Performance: We compare our approach with top entries on Argoverse motion forecasting leaderboard [26] in Tab. 5.3. SSL-Lanes improves the metrics for $K = 1$ convincingly and outperforms existing approaches w.r.t. min-ADE$_1$, min-FDE$_1$ and MR$_1$. We are also strongly competitive w.r.t. min-ADE$_6$, min-FDE$_6$ and MR$_6$ against top approaches, with a relatively simple architecture.

Trade-off between min-FDE and Miss-Rate: min-FDE$_6$ and MR$_6$ are both important for autonomous robots to optimize. Ideally we wish for both of these metrics to be low. However, there exists a frequent trade-off between them. We compare this trade-off in Fig. 5.2(a) with six other popular motion forecasting models (in terms of citations and GitHub stars), namely: Lane-GCN [88], Lane-RCNN [88], MultiPath [25], mm-Transformer [98], TNT [205] and Dense-TNT [51] on the Argoverse validation set. We are on the lowest-left of meaning we optimize both min-FDE$_6$ and MR$_6$ successfully in comparison to other top models.

Trade-off between accuracy, efficiency and complexity: We are the first to point out a trade-off that exists for current state-of-the-art motion forecasting models

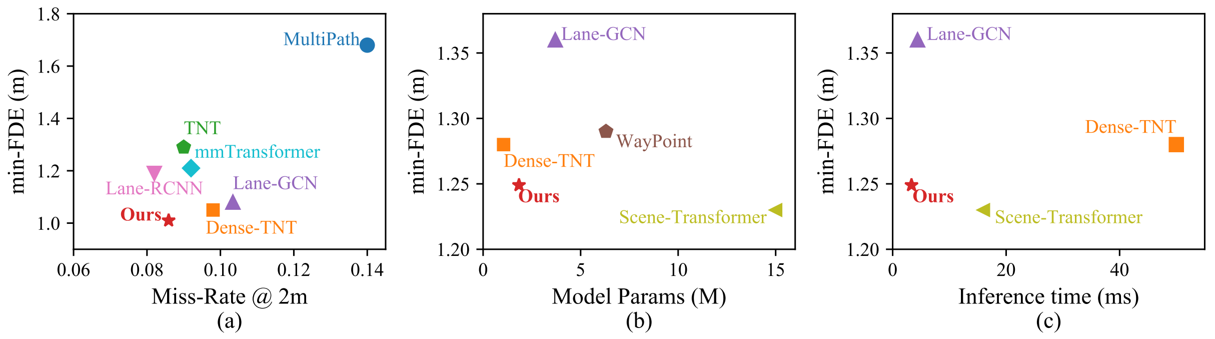


Figure 5.2: *Left:* min-FDE$_6$ - Miss-Rate$_6$ trade-off on Argoverse Validation. Lower-left is better. We optimize both successfully in comparison to other popular approaches. *Right:* We plot min-FDE on Argoverse Test Set against number of model parameters (in millions) and inference time (in milli-seconds). We find that there is a trade-off between min-FDE performance, architectural complexity (as measured by number of parameters) and computational efficiency (as measured by inference time). Our work achieves the best trade-off (lower-left).

between forecasting performance, architectural complexity and inference speed. This is illustrated in Fig. 5.2(b)-(c). NN+Map [26] (see Tab. 5.3) is a simple nearest-neighbor based approach that also uses map-features, and while it has advantages in terms of fast inference and low model complexity, the forecasting performance is very low. MultiPath [25] is a very popular approach that has reasonable accuracy and inference speed but is parametrically heavy due to it's use of convolutional kernels. Lane-GCN is a vector based approach [88] has comparatively fast inference time and high accuracy, but uses multiple GNN layers which can lead to problems with over-smoothing for map-encoders [133] and also has a complicated four-stage fusion mechanism. Lane-RCNN [197] proposes to capture interactions between agents and map using not just a single vector, but a local interaction graph per agent - this adds huge number of hyper-parameters to the model and makes it very complex. Transformer-based models [98, 66, 111, 47] also suffer in this regard. Scene-transformer for example has 15M parameters and uses heavy augmentation to prevent overfitting. A light high-performing model is Dense-TNT [51]. However, Dense-TNT's inference speed on average is 50ms per agent, because it proposes a time-intensive optimization algorithm to find a dense goal set that minimizes the expected error of the given set. In contrast to these popular models, our approach has high accuracy (min-FDE: 1.25m, MR: 13.3%) while also having low architectural complexity (1.84M parameters) and high inference speed (3.30 ms). Thus it provides a great balance for application to

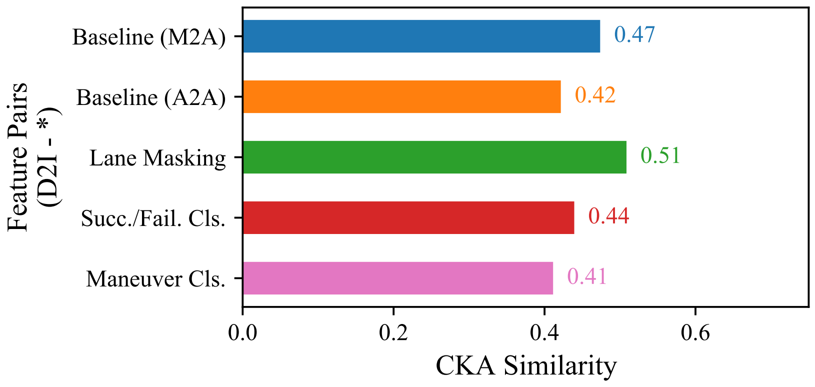


Figure 5.3: CKA Feature similarity between feature pairs of baseline and different pretext tasks. Similarity score is 1 for completely overlapping features and 0 for completely divergent features.

real-time safety-critical autonomous robots.

Qualitative Results: We present some multi-modal prediction trajectories on several hard cases shown in Fig. 5.4. The yellow trajectory represents the observed 2s. Red represents ground truth for the next 3s and green represents the multiple forecasted trajectories for those 3s. In Row 1, the agent turns right at the intersection. The baseline misses this mode completely, despite having access to the map. The model trained with lane-masking successfully predicts this right turn within 2m of the ground-truth end-point. In Row 2, the agent has a noisy past history and accelerates while turning left at the intersection. The pretext task distance-to-intersection can correctly capture this, while the baseline has only one trajectory covering this mode but vastly overshoots the ground-truth. Interestingly, we note that the success/failure pretext task is unable to capture this mode. We believe this is due to a stronger prior imposed by the model during learning. In Row 3, we have an agent accelerating while going straight at an intersection. We find that the maneuver classification pretext task is the only model that correctly predicts trajectories aligned with the ground-truth. In Row 4, we have an agent turning left at an intersection. Most of the predictions of other models predicts that the agent will go straight. The success/failure pretext task however picks up on the left-turn, possibly due to the priors imposed upon it by end-point conditioning.
Overall, SSL-Lanes can capture left and right turns better, while also being able to discern acceleration at intersections. Our pretext tasks provide priors for the model and provides data-regularization for free. We believe this can improve forecasting through better under-

Description	**Experimental Setup** **Training**	**Validation**	**Method**	**minADE$_6$**	**minFDE$_6$**	**MR$_6$**
Effects of limited training data	25% of train	All	Baseline	0.82	1.33	14.66
			Ours	**0.78**	**1.22**	**12.63**
Effects of new domain	100% PIT + 20% MIA	MIA val	Baseline	0.88	1.46	17.21
			Ours	**0.85**	**1.34**	**14.96**
Performance on difficult maneuvers	All	Turning & lane changing	Baseline	0.90	1.53	19.90
			Ours	**0.84**	**1.34**	**14.93**
Effects of imbalanced data	2x straight 1x other maneuvers	Turning & lane changing	Baseline	0.94	1.65	21.53
			Ours	**0.90**	**1.49**	**17.97**
Effects of noisy data	All	Gaussian noise ($\sigma = 0.2$) with $p = 0.25$	Baseline	1.01	1.37	15.59
			Ours	**0.96**	**1.24**	**11.98**
Effects of noisy data	All	Gaussian noise ($\sigma = 0.2$) with $p = 0.5$	Baseline	1.19	1.56	20.64
			Ours	**1.13**	**1.40**	**15.65**

Table 5.4: Different experimental settings to provide evidence for why SSL-based training helps motion forecasting

standing of map topology, agent context with respect to the map, and generalization with respect to imbalance implicitly present in data.

5.6.3 When does SSL help Motion Forecasting?

Hypotheses: We hypothesize that training with SSL pretext tasks probably helps motion forecasting as following: (a) Topology-based context prediction assumes feature similarity or smoothness in small neighborhoods of maps. Such a context-based feature representation can greatly improve prediction performance, especially when the neighborhoods are small. (b) Clustering and classification assumes that feature similarity implies target-label similarity and can group distant nodes with similar features together, leading to better generalization. (c) Supervised learning with imbalanced datasets sees significant degradation in performance. Although most of the data samples in Argoverse are at an intersection, a significantly large number involve driving straight while maintaining speed. Recent studies [97] have shown that SSL tends to learn richer features from more frequent classes which also allows it to generalize to to other classes better.

Experiments: In order to provide evidence for our hypotheses, we propose to design 6 different training and testing setups as shown in Tab. 5.4. We use success/failure classification as the pretext task, and all models are trained for 50,000 steps. We initialize the map-encoder with the parameters from a model trained with the lane-masking pretext

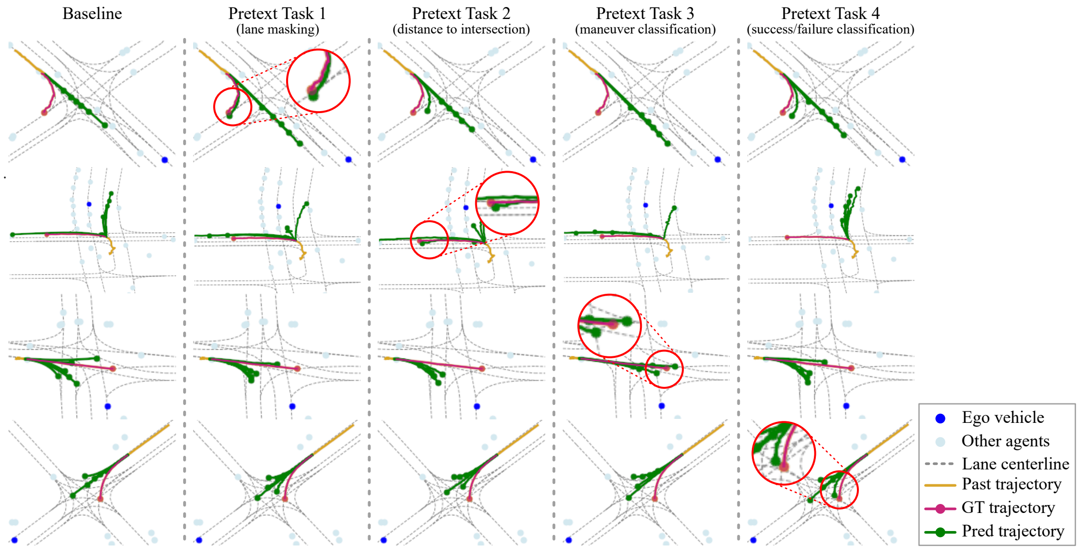


Figure 5.4: Motion forecasting on Argoverse [26] validation. We show four challenging scenarios at intersections. The baseline [88] misses all the predictions. In the first row, our proposed lane masking successfully captures the right-turn. For the second row, predicting distance to intersection helps the most in capturing the left turn. In the third row, acceleration at an intersection is best captured by the model that is made to classify maneuvers of traffic agents. Finally, in the fourth row, classifying successful final goal states is the most effective at capturing the left turn. These tasks are trained with pseudo-labels which are obtained for free from data. Please refer to Sec. 5.6.2 for details.

task.

Our *first* setting is to train with 25% of the total data available for training and testing on the full validation set. We expect the SSL-based task to capture richer features and generalize better than the baseline. Our *second* setting assumes that SSL also generalizes to topology from different cities and trains on 100% of data from Pittsburgh (PIT) but only 20% of data from Miami (MIA). For evaluation, we only test on data examples taken from the city of MIA. For our *third* setting, we assume that SSL learns superior features and can thus perform better in difficult cases like lane-changes and turning cases. For evaluation, we only test on data examples which involves these difficult cases. In our *fourth* setting, we choose to explicitly train with data that contains 2× ‘straight-with-same-speed’ maneuver and 1× all other maneuvers. We test only on lane-changes and turning cases from validation. Finally in order to test the effect of noise on motion forecasting performance, we take two models already trained on full data. We now take the full validation set, randomly select agent trajectories or map nodes with probability $p = 0.25$ and $p = 0.5$, and then add Gaussian noise with zero mean and 0.2 variance to their features. We expect this to have the most impact on forecasting performance as compared to all other settings since this is the most aggressive form of corruption. But we also expect SSL-based pretext task training to provide robustness to noise for free due to better generalization capabilities. Our takeaway from these experiments is that there is strong evidence SSL-based tasks do provide better generalization capabilities and can thus prove to be more effective than pure supervised training based approaches.

5.7 Chapter Conclusion

We propose SSL-Lanes to leverage supervisory signals generated from data for free in the form of pseudo-labels and integrate it with a standard motion forecasting model. We design four pretext tasks that can take advantage of map-structure and similarities between agent dynamics to generate these pseudo-labels, namely: lane masking, distance to intersection prediction, maneuver classification and success/failure classification. We validate our proposed approach by achieving competitive results on the challenging large-scale Argoverse benchmark. The main advantage of SSL-Lanes is that it has high accuracy combined with low architectural complexity and high inference speed. We further demonstrate that each proposed SSL pretext task improves upon the baseline, especially in difficult cases like left/right turns and acceleration/deceleration. We also provide hypotheses and experiments on why SSL-Lanes can improve motion forecasting.

In the upcoming chapter, Chapter 6, we focus on pretext tasks to enhance interaction modeling, particularly in scenarios where agent interaction plays a significant role.

Chapter 6

SSL-Interactions: Pretext Tasks for Interactive Trajectory Prediction

Autonomous vehicle safety is contingent on effective multi-agent prediction. Previously, we presented the SSL-Lanes framework in Chapter 5, utilizing pretext tasks to enhance motion forecasting. Despite its merits, this framework overlooks interaction-specific domain knowledge, crucial for improving interactive trajectory prediction. This chapter introduces the SSL-Interactions framework, which integrates specially designed pretext tasks to advance interaction modeling in trajectory prediction. These tasks, rich in interaction-specific domain knowledge, not only facilitate effective model regularization, but also provide a balanced and computationally efficient solution between marginal and fully-joint predictions. We will examine the design and practical application of these interaction-aware pretext tasks within the SSL-Interactions framework. SSL-Interactions code and documentation is open-sourced.[1]

6.1 Motivation and Contributions

By accurately predicting the future trajectories of other agents in a dynamic environment, autonomous vehicles can strategically plan and execute maneuvers without collisions. However, in a multi-agent environment, numerous entities simultaneously interact and influence each other's behaviors. The main challenge lies in accurately capturing these complex in-

[1]https://github.com/AutoVision-cloud/SSL-Interactions

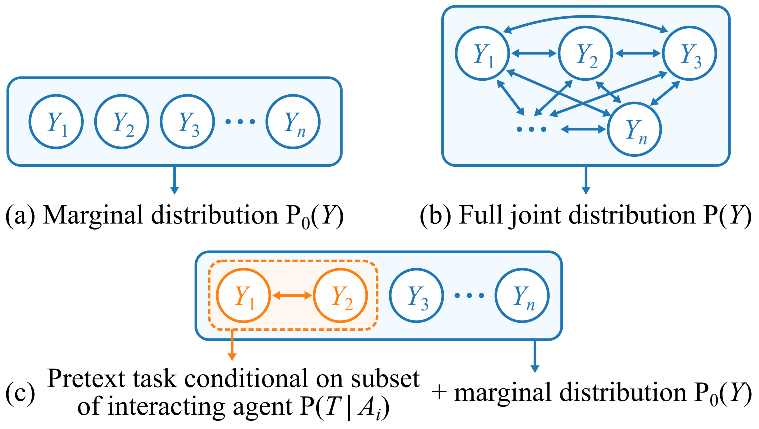

Y_1
Y_2
Y_3
Y_n
(a) Marginal distribution $P_0(Y)$
(b) Full joint distribution $P(Y)$
(c) Pretext task conditional on subset of interacting agent $P(T \mid A_i)$
+ marginal distribution $P_0(Y)$

Figure 6.1: Comparison of approaches for training multi-agent forecasting systems. (a) *Interactions between agents are not considered explicitly.* Each node represents an agent's state, and arrows denote the aggregation of these marginal distributions $\mathrm{P}_0(\boldsymbol{Y})$. (b) *Interactions between all pairs of agents are considered.* The nodes represent the random variables, and the bidirectional edges between every pair of nodes denote the dependencies among all pairs of random variables. $P(\boldsymbol{Y})$ indicates the estimation of the full joint interaction distribution. (c) *SSL-Interactions considers interactions between pseudo-labeled pairs of agents.* The nodes represent the random variables, grouped by interaction-specific dependencies in subset A_i enclosed by dashed-line rectangles. $P(\boldsymbol{T}|A_i)$ represents the conditional distributions for pretext task. This is used to train the interaction-module in a self-supervised setup, and drives $\mathcal{L}_{pretext}$ in Fig. 6.3. $P_0(\boldsymbol{Y})$ captures the marginal distributions of all the agents. This is used to train the forecasting framework similar to the approach in (a), and drives $\mathcal{L}_{main}$ in Fig. 6.3.

teractions and developing models that can account for the coordination and cooperation between various agents.

In this chapter, we propose utilizing pretext tasks as a method for improved interaction modeling in motion forecasting. This approach offers several advantages. Firstly, pretext tasks offer a way to decompose the joint distribution into a conditional and a marginal, presenting a computationally feasible yet effective solution for training the interaction module as shown in Fig. 6.1. Secondly, pretext tasks enable the incorporation of interaction-specific domain knowledge during the learning process. By designing tasks that reflect meaningful aspects of agent interaction, we can guide the model to capture relevant patterns in the data, which acts as an effective regularization technique. This helps enhance the model's understanding of the underlying dynamics and improves the model's accuracy in forecasting agent motion. Thirdly, pretext tasks can be designed in a manner that allows pseudo-labels to be generated from the available data itself. This enables us to leverage large amounts of unlabeled data, which can be invaluable in training the model effectively.

We introduce four interaction-aware pretext tasks for motion forecasting: range gap prediction, closest distance prediction, direction of movement prediction, and type of interaction prediction. These pretext tasks are designed to capture specific aspects of agent interaction, improving the model's understanding of complex scenarios. In addition, we propose an approach to curate interactive scenarios from a large dataset and explicitly pseudo-label interaction pairs. This curation process facilitates pretext task learning and adds to the model's ability to capture meaningful interactions.

To evaluate the effectiveness of our proposal, we introduce three metrics that provide a better assessment of predictions in interactive scenarios. These metrics include interactive min-FDE(i-min-FDE), non-interactive min-FDE(ni-min-FDE), and collision awareness metric(CAM). We also demonstrate the benefits of SSL-Interactions over the state-of-the-art, in terms of quantitative metrics and qualitative analysis.

Contributions: The contributions of this work are threefold:

- We propose a framework called SSL-Interactions, that leverages pretext tasks to improve interaction modeling for motion forecasting. Specifically, we develop four pretext tasks, designed to capture various aspects of interaction based on domain-specific knowledge.
- The majority of recent motion forecasting datasets do not clearly identify pairs of interacting agents, relying instead on the implicit interaction modeling conducted via end-to-end training [88, 98]. We propose a simple but effective way to curate interaction-specific scenarios from datasets and to explicitly label pairs of interacting agents within a given scenario. This approach is crucial for generating pseudo-labels for interaction-based pretext tasks.
- Through empirical evaluation, we demonstrate that our proposed framework can surpass a state-of-the-art motion forecasting method both quantitatively, with up to an 8% improvement, and qualitatively. Furthermore, we introduce three new metrics specifically designed to evaluate predictions within interactive scenes.

6.2 Problem Formulation

6.2.1 Notations

We are given as input a dataset of past movements for N actors. Each actor is represented by a set of (x, y) coordinates indicating their center locations during the preceding $T_p - 1$ time steps. To prepare the data for analysis, we preprocess each trajectory into a sequence of displacements denoted by $\mathcal{S}_i = \{\Delta \boldsymbol{s}_i^{-\{T_p-1\}+1}, ..., \Delta \boldsymbol{s}_i^{-1}, \Delta \boldsymbol{s}_i^{0}\}$. Here, $\boldsymbol{s}_i^{t_p}$ represents the 2D displacement from the time-step $t_p - 1$ to t_p. This pre-processing ensures that the model concentrates on forecasting relative coordinates rather than absolute positions. The set of past motions of all N actors is given by $\mathcal{M} = \{\mathcal{S}_1, ..., \mathcal{S}_N\}$. Additionally, we have access to a high-definition (HD) map that includes lane and semantic attributes. We use a scene-centric embedding where we use an agent of interest's position as the origin, and encode all roadgraph and agents with respect to it. The map elements are encoded into an embedding $\boldsymbol{h}_0$.

The ground-truth future motion of each actor in the scene is also provided and denoted by $\mathcal{Y}_i^{GT} = \{(x_i^1, y_i^1), ..., (x_i^{T_c}, y_i^{T_c})\}$ over a prediction horizon of T_c. The set of ground-truth futures is given by $\mathcal{O}_{\text{GT}} = \{\mathcal{Y}_i^{GT} | i = 1, ..., N\}$. The future trajectory prediction for agent i is represented by $\mathcal{Y}_i = \{(\hat{x}_i^1, \hat{y}_i^1), ..., (\hat{x}_i^{T_c}, \hat{y}_i^{T_c})\}$ and $\mathcal{O}_{\text{pred}} = \{\mathcal{Y}_i | i = 1, ..., N\}$. But there could be multiple feasible future predictions for a given past input. Our goal is to predict K possible future trajectories, $\mathcal{P}_F$, where $\mathcal{P}_F = \{\mathcal{O}_{\text{pred, 1}}, \mathcal{O}_{\text{pred, 2}}, ..., \mathcal{O}_{\text{pred, K}}\}$. We assume that the agents interacting in a given scene are not provided with specific interaction labels. Thus we want a trajectory prediction model that learns to model the distribution $p(\boldsymbol{\mathcal{Y}} | \mathcal{M}, \boldsymbol{h}_0)$.

6.2.2 Standard Interaction Modeling

The encoder for the input motion of the agents is parameterized by f_{enc}, and the encoding can be represented by Eq. (6.1). To incorporate contextual information such as lane semantics, a graph convolutional neural network (GCN) parameterized by f_{map}, is employed to operate on a graph representation of the scene depicted in Eq. (6.2) and called a M2A layer. Finally, to model agent-to-agent interactions, the representation of nearby agents is aggregated using a graph convolutional neural network (GCN) parameterized by f_{interact}. This is called an A2A layer. The graph representation is constructed such that agents are represented as nodes in the graph, and edges are established between agents if they are within a certain distance of each other. Specifically, the neighboring set of node i is represented as $\mathcal{N}_e$, and the distance between agents i and j is given by d_{ij}. The input to the GCN includes the agent i's embedding $\hat{\boldsymbol{v}}_i$, the contextual embedding obtained for each neighboring agent $\{\hat{\boldsymbol{v}}_j | j \in \mathcal{N}_e\}$, and d_{ij}. During the GCN operation, the interaction information from neighboring agent states is *implicitly* incorporated by performing message passing between nodes in the graph, as depicted in Eq. (6.3). By incorporating the temporal representation of nearby agents and their relative distances into the final agent state $\tilde{\boldsymbol{v}}_i$ to be used for future prediction, agent-to-agent interactions are modeled within the data-driven motion forecasting framework.

$$\hat{\boldsymbol{s}}_i = f_{\text{enc}}(\mathcal{S}_i) \tag{6.1}$$

$$\hat{\boldsymbol{v}}_i = f_{\text{map}}(\hat{\boldsymbol{s}}_i, \boldsymbol{h}_0) \tag{6.2}$$

$$\tilde{\boldsymbol{v}}_i = \sum_{j \in \mathcal{N}_e} f_{\text{interact}}(\hat{\boldsymbol{v}}_i, \hat{\boldsymbol{v}}_j, d_{ij}) \tag{6.3}$$

6.2.3 Pretext Task Regularization

Pretext task regularization is a technique that involves training a model on a auxiliary task, called a pretext task, in order to improve the model's performance on the main task. The idea is to leverage the structure of the pretext task to encourage the model to learn representations that are transferable to the main task of interest [91]. A pretext task is designed using domain-knowledge to capture some inherent structure or pattern in the data. The most effective pretext tasks exploit the underlying structure in such a way that the learned representations are meaningful and generalizable. This does not add to model complexity, as the pretext tasks parameters are discarded at inference time.

The existing literature for trajectory prediction predominantly focuses on supervised learning scenarios where the learner has access to a large, annotated dataset that can be revisited multiple times to learn the optimal feature extractor f_θ. However, in numerous real-world scenarios, we often have access to vast amounts of unlabeled data, which can potentially make the learning process more efficient and cost-effective. In this context, our objective is to investigate if designing pretext tasks that emphasize the semantics of inter-object interactions can lead to more meaningful representations for agents, ultimately enhancing the model's ability to make accurate future predictions.

To incorporate a pretext task into the learning process, we train a model on both the main task and the pretext task simultaneously. Let the model parameters be denoted by $\boldsymbol{\Theta}$. Let the main task loss function be $L_{\text{main}}(\boldsymbol{\Theta})$, and the pretext task loss function by

$L_{\text{pretext}}(\boldsymbol{\Theta})$. The overall loss function for the combined learning problem can be expressed as Eq. (6.4). λ is a regularization hyper-parameter that controls the balance between the main task and the pretext task. The model's parameters are updated using gradient-based optimization.

$$L_{\text{total}}(\boldsymbol{\Theta}) = L_{\text{main}}(\boldsymbol{\Theta}) + \lambda * L_{\text{pretext}}(\boldsymbol{\Theta}) \tag{6.4}$$

In summary, pretext tasks provide a powerful framework for leveraging the structure of unlabeled data to learn useful representations that can be transferred to the main task.

6.3 SSL-Interactions

We introduce SSL-Interactions, a trajectory prediction framework that leverages self-supervised learning to capture social interaction from our designed pretext tasks. SSL-Interactions trains on both the main task of trajectory prediction and the self-supervised pretext tasks simultaneously, using the latter to augment the model's ability to capture relevant features. The overall architecture is illustrated in Fig. 6.3.

Advantages: Unlike transformers, which require large amounts of data to learn effectively, SSL-Interactions can be trained on smaller datasets by leveraging the pretext tasks. By generating self-supervised labels related to the downstream task, we can use a larger amount of training data to train the data representation, rather than relying solely on the downstream task's labels. Additionally, this approach ensures that the model learns useful features relevant to the main task, while benefiting from the additional self-supervised data.

6.3.1 Design Considerations for Interaction Modeling

(A) Marginal Prediction: In general, a trajectory prediction model learns to model the distribution $p(\boldsymbol{\mathcal{Y}}|\mathcal{M}, \boldsymbol{h}_0)$. In the case of multiple agents, trajectory prediction is often evaluated independently for each agent. This means that accurately predicting the marginal distribution of vehicle trajectories is sufficient for achieving good results on benchmark tasks. Although the agent-to-agent (A2A) layer *implicitly* models interactions within the framework, the training parameters may focus solely on optimizing the final forecasting loss. In Sec. 6.2.2, we consider a representative motion forecasting model of this type. This model captures interactions between agents implicitly, using message passing via graph convolutions to model $p(\mathcal{Y}_i|\mathcal{M}, \boldsymbol{h}_0)$. However, models that focus solely on predicting marginal distributions may generate joint behaviors that are infeasible or unrealistic.

(B) Full Joint Prediction: To properly evaluate and assess such interactive behaviors, it is necessary to predict the joint distribution of the future trajectories of all interacting agents. Directly modeling the joint distribution $p(\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_N | \mathcal{M}, \boldsymbol{h}_0)$ however grows exponentially with the number of agents. This presents a significant challenge for models that aim to accurately predict joint behaviors for complex interactions between multiple agents.

(C) SSL-Interactions: We propose to use pseudo-labeled interacting pairs to train the interactions *explicitly* via pretext tasks. A pretext task is defined as T. Each pretext task T is associated with a subset of agents $A_i \in \{\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_{k_i}\}$, where $k_i < N$ is the number of agents involved in task T for agent i. Under the assumption that the pretext tasks capture important conditional independence properties among the agents, we can factorize the joint probability distribution $p(\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_N | \mathcal{M}, \boldsymbol{h}_0)$ into a product of a marginal distribution $p_0(\mathcal{Y}_i | \mathcal{M}, \boldsymbol{h}_0)$ and pretext task-specific conditional distributions $P(T|A_i)$. This is shown in Eq. (6.5).

$$p(\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_N | \mathcal{M}, \boldsymbol{h}_0) = \prod_{i=1}^{N} p_0(\mathcal{Y}_i | \mathcal{M}, \boldsymbol{h}_0) \prod_{i=1}^{N} P(T|A_i) \tag{6.5}$$

$p_0(\mathcal{Y}_i | \mathcal{M}, \boldsymbol{h}_0)$ is a distribution that captures the marginal dependencies among the agents as described in Sec. 6.3.1(A), and drives $\mathcal{L}_{main}$ in Fig. 6.3. $P(T|A_i)$ is the conditional probability distribution of pretext task T given the states of the agents in A_i, which captures interaction information between them *explicitly*, and drives $\mathcal{L}_{pretext}$ in Fig. 6.3.

The factorization in Eq. (6.5) reduces the complexity of the joint distribution by decomposing it into simpler components. To model each component, one can employ maximum likelihood estimation. For the our proposed pretext tasks Sec. 6.3.3, the negative log-likelihood loss for the pretext tasks corresponds to a Mean Squared Error (MSE) loss for regression setting when we incorporate the Gaussian likelihood assumption, and cross-entropy loss for the classification setting.

Identifying the optimal decomposition and pretext tasks for a given problem can be a challenging task, and may require domain-specific knowledge. In this chapter, we propose both pretext tasks tailored for interactive trajectory prediction, and a method to pseudo-label the optimal subset of interacting agents A_i.

6.3.2 Labeling Interactions

In Sec. 6.2, our assumption is that agents in a given scene do not have access to explicit interaction labels. To train a predictor that utilizes pretext tasks, we first produce a dataset

Algorithm 2 Label Interacting Trajectories

```
1: procedure LabelInteractingTrajectories($\mathcal{T}$)
2:     Input: Scene Trajectories $\mathcal{T} = \{\mathcal{Y}_1^{GT}, ..., \mathcal{Y}_n^{GT}\}$
3:     Output: Filtered interacting pairs $\mathcal{I}$
4:     for each pair of trajectories $(\mathcal{Y}_i^{GT}, \mathcal{Y}_j^{GT})$ do
5:         Calculate $d_{ij} = min(||p_i^{t_1} - p_j^{t_2}||)$ as Eq. (6.6)
6:         if $d_{ij} < d_{th}$ then
7:             Mark pair $(\mathcal{Y}_i^{GT}, \mathcal{Y}_j^{GT})$ as interacting
8:         end if
9:     end for
10:    for each target agent $t$ do
11:        Classify target agent's intent as $I_t$
12:    end for
13:    for each interacting pair $(\mathcal{Y}_i^{GT}, \mathcal{Y}_j^{GT})$ do
14:        if $I_t \in \{$'Left-Turn', 'Left-Turn-Waiting'$\}$ then
15:            Keep oncoming agents
16:        else
17:            Remove oncoming agents
18:        end if
19:    end for
20:    return Filtered interacting pairs $\mathcal{I}$
21: end procedure
```

that contains explicit interaction labels. Specifically, we wish to label A_i for agent i, as described in Sec. 6.3.1.

To ensure effective interaction modeling, we visually query a random subset of the data and discover several limitations. In some cases, the target agent is the only vehicle present in a large area around it, limiting the potential for interaction with surrounding vehicles. In other cases, the target agent cannot be influenced by surrounding vehicles due to factors such as distance or the absence of spatio-temporal conflict. It is clear that a model trained solely on this data would not contribute significantly to interaction modeling. Therefore, to improve our model's ability to capture interactions, it is necessary to curate a dataset that specifically addresses these limitations and includes instances where meaningful interactions occur.

We propose a simple but effective approach to identify interacting pairs of trajectories based on the spatial distance between the agents and their intents. The method consists

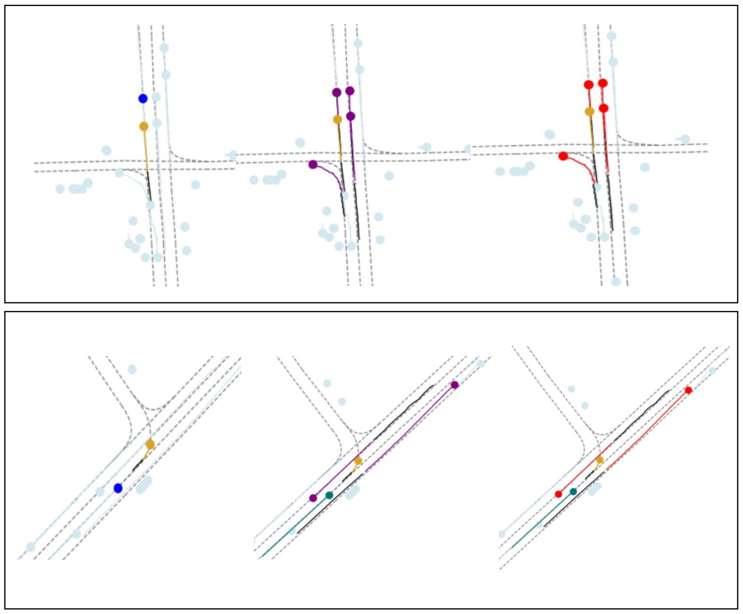


Figure 6.2: Illustration of the proposed data curation method for explicitly labeling pairwise interactions. The future trajectory of the target agent is denoted in yellow, while the past inputs are given in black. The first step involves identifying agents within a specified distance threshold, indicated by the violet color. Nonetheless, only distance thresholding is inadequate, as vehicles moving in opposite directions frequently do not interact. In the second step, oncoming agents are filtered out if the target agent intends to proceed straight, but are retained if the target's intended action is a left turn. The final interacting agents' future trajectories are given in red.

of three main steps, as described below. We do not label trajectories if they are too short.

- *Defining interaction between a pair of trajectories:* We define two trajectories, $\mathcal{Y}_i^{GT}$ and $\mathcal{Y}_j^{GT}$, as interacting if the closest spatial distance between the two agents at any time point in the ground-truth (GT) future is less than a threshold distance, $d_{th} = 5$m. Mathematically, we are given trajectories $\mathcal{Y}_i^{GT} = \{p_i^1, p_i^2, ..., p_i^{T_c}\}$ and $\mathcal{Y}_j^{GT} = \{p_j^1, p_j^2, ..., p_j^{T_c}\}$. $||p_i^{t_1} - p_j^{t_2}||$ denotes the Euclidean distance between point $p_i^{t_1}$ from trajectory $\mathcal{Y}_i^{GT}$ and point p_j^{t2} from trajectory $\mathcal{Y}_j^{GT}$. For time points $t_1 \in \{1, ..., T_c\}$ and $t_2 \in \{1, ..., T_c\}$, d_{ij} is given by:

$$d_{ij} = min(||p_i^{t_1} - p_j^{t_2}||) \tag{6.6}$$

  The two agents i and j are interacting if $d_{ij} < d_{th}$. This is Step-1.

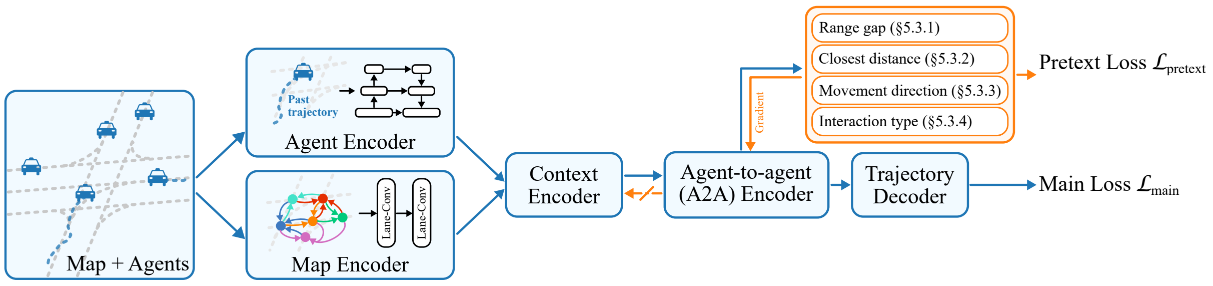

Map + Agents
Past trajectory
Agent Encoder
Lane-Conv
Lane-Conv
Map Encoder
Context Encoder
Agent-to-agent (A2A) Encoder
Gradient
Range gap (§5.3.1)
Closest distance (§5.3.2)
Movement direction (§5.3.3)
Interaction type (§5.3.4)
Pretext Loss $\mathcal{L}_{\text{pretext}}$
Trajectory Decoder
Main Loss $\mathcal{L}_{\text{main}}$

Figure 6.3: Schematic diagram of the proposed model incorporating a pretext task as an auxiliary loss. The agent encoder processes each agent's observed trajectory, while the map encoder handles high-definition (HD) map encoding. These representations are initially passed to a context encoder, generating map-conditioned agent features. These are subsequently processed by an agent-to-agent attention-based encoder, which encodes inter-agent dependencies. This comprehensive representation informs both the future trajectory decoder and the proposed pretext task component. The pretext task loss, benefiting from a stop gradient, exclusively trains the agent-to-agent encoder, ensuring only interaction-specific features are harnessed by the pretext tasks. Consequently, any improvements can be specifically attributed to enhanced interaction modeling within the agent-to-agent encoder. The pretext task loss, serving as an auxiliary task, is discarded during the inference phase.

- *Classifying target agent intent:* Given a target agent, proximity does not guarantee interaction. For example: if the target agent is going straight, an oncoming agent in a different lane is unlikely to contribute to its future motion. However, if the target agent is turning left, then the oncoming agent is likely to contribute to its future motion. We thus propose a heuristic conditioned on the target agent intent to filter out agents identified by Step-1. Given the trajectory information of a target agent i, we classify its intent into six categories: {Straight, Lane-Change, Right-turn, Left-turn, Right-turn-Waiting, Left-turn-waiting, Other}. This is Step-2.

- *Filtering agents based on target agent's intent:* In this step, we use the target agent's intent from Step-2 to filter out spatially close agents that are not interacting. We propose the following heuristic: if the target agent's intent belongs to either 'Left-turn' or 'Left-turn-waiting', we retain oncoming agents; otherwise, we exclude oncoming agents. In summary, this step scrutinizes the target agent's intent to ascertain whether to retain or exclude oncoming agents, effectively filtering out non-interacting agents in close proximity. This is Step-3.

The overall algorithm is given in Algorithm 2. For every scene, we simplify our problem to filter only $(\mathcal{Y}^{GT}_{\text{target}}, \mathcal{Y}^{GT}_{j})$. This is because datasets like Argoverse have already pre-selected the target agent of interest, and we can focus our attention solely on the interactions between the target agent and the other agents in the scene. Thus A_i for agent i is the output of Algorithm 2. A visual illustration of the data curation process is given by Fig. 6.2.

6.3.3 Proposed Pretext Tasks

In this section, we present four pretext tasks for interaction and motion modeling: 1) range-gap prediction, 2) closest distance prediction, 3) direction of movement prediction and 4) type of interaction prediction. The overall setup of the pretext task is given by Fig. 6.4.

A. Range-gap Prediction

In this pretext task, the main idea is to predict the range-gap between pairs of agents at a future time step $t = 2s$, given their past trajectory information. The range-gap can be defined as the difference in the distance traveled by two agents during a specific time interval. This task focuses on learning the interaction patterns between agents by considering the spatial and temporal correlations in their trajectories.

Formulation: Let the pretext task feature extractor for this task to be denoted by f_{RG} with the parameters $\boldsymbol{\Theta}_{\text{RG}}$. We consider the ground-truth range gap between two agents i which is the target agent and j to be $||p^{t_1}_{\text{target}} - p^{t_2}_{j}||$ at time $t_1 = 2s$ and $t_2 = 2s$.
The auxiliary range gap feature predictor takes the following inputs: the agent-to-agent interaction feature from the standard motion forecasting architecture as described in Sec. 6.2.2 represented by $\tilde{\boldsymbol{v}}_{\text{target}}$ and $\tilde{\boldsymbol{v}}_{j}$, and the distance between them at $t_p = 0$ given by $d^{0}_{\text{target}:j}$. The output is a predicted range gap distance between the target agent and j.

$$d^{\text{RG}}_{j} = f_{\text{RG}}(\tilde{\boldsymbol{v}}_{\text{target}} - \tilde{\boldsymbol{v}}_{j}, d^{0}_{\text{target}:j}) \tag{6.7}$$

The objective function $\boldsymbol{\mathcal{L}}_{\text{RG}}$ for guiding the range-gap prediction is formulated using Smooth-L1 loss as shown in Eq. (6.8). k_{target} is the number of interacting agents with respect to the target agents as obtained from Sec. 6.3.2.

$$\boldsymbol{\mathcal{L}}_{\text{RG}}(\boldsymbol{\Theta}_{\text{RG}}) = \frac{1}{k_{\text{target}}} \sum_{j=1}^{k_{\text{target}}} L_{\text{reg}}(d^{\text{RG}}_{j}, ||p^{t_1}_{\text{target}} - p^{t_2}_{j}||) \tag{6.8}$$

Benefit to final forecasting performance: By predicting range-gaps, the model learns to represent and capture the dependencies between the agents' movements more effectively. For example, if the range gap predicted between two agents is low, it could indicate that the two agents are getting closer to each other and may potentially collide. Conversely, if the range gap predicted is high, it could indicate that the two agents are moving away from each other and may not interact. This helps the model understand the underlying structure and patterns of agent interactions better, ultimately leading to more accurate motion forecasts.

B. Closest-distance Prediction

In this pretext task, the main idea is to predict the minimum distance between two agents in the future. In other words, we directly predict the closest distance that will occur between the agents at corresponding time steps in the T_c time step future, without explicitly analyzing the distance at each future time step. This task enables the model to acquire meaningful representations of agent interactions in dynamic environments.

Formulation: Let the pretext task feature extractor for this task to be denoted by f_{CD} with the parameters $\mathbf{\Theta}_{\text{CD}}$. We consider the ground-truth closest distance between two agents i, which is the target agent, and j to be calculated as: $D_j^{\text{GT}} = \min\{D_{\text{target}:j}(t) | t \in 1, ..., T_c\}$, where, $D_j^{\text{target}:j}(t) = ||p_{\text{target}}^t - p_j^t||$. Additionally, we want to setup this task as a classification task instead of a regression task. Thus we choose four classes with labels y_j^{CD}:

$$y_j^{\text{CD}} = \begin{cases} 0 & \text{if} \quad 0 < D_j^{\text{GT}} \leq 5 \\ 1 & \text{if} \quad 5 < D_j^{\text{GT}} \leq 10 \\ 2 & \text{if} \quad 10 < D_j^{\text{GT}} \leq 15 \\ 3 & \text{if} \quad D_j^{\text{GT}} > 15 \end{cases} \tag{6.9}$$

The auxiliary closest distance feature predictor takes the following inputs: the agent-to-agent interaction feature from the standard motion forecasting architecture as described in Sec. 6.2.2 represented by $\tilde{\boldsymbol{v}}_{\text{target}}$ and $\tilde{\boldsymbol{v}}_j$, and the distance between them at $t_p = 0$ given by $d_{\text{target}:j}^0$. The output is a predicted closest distance between the target agent and j.

$$d_j^{\text{CD}} = \text{Softmax}(f_{\text{CD}}(\tilde{\boldsymbol{v}}_{\text{target}} - \tilde{\boldsymbol{v}}_j, d_{\text{target}:j}^0)) \tag{6.10}$$

The objective function $\boldsymbol{\mathcal{L}}_{\text{CD}}$ for guiding the closest distance prediction is formulated using cross-entropy loss as shown in Eq. (6.11). k_{target} is the number of interacting agents

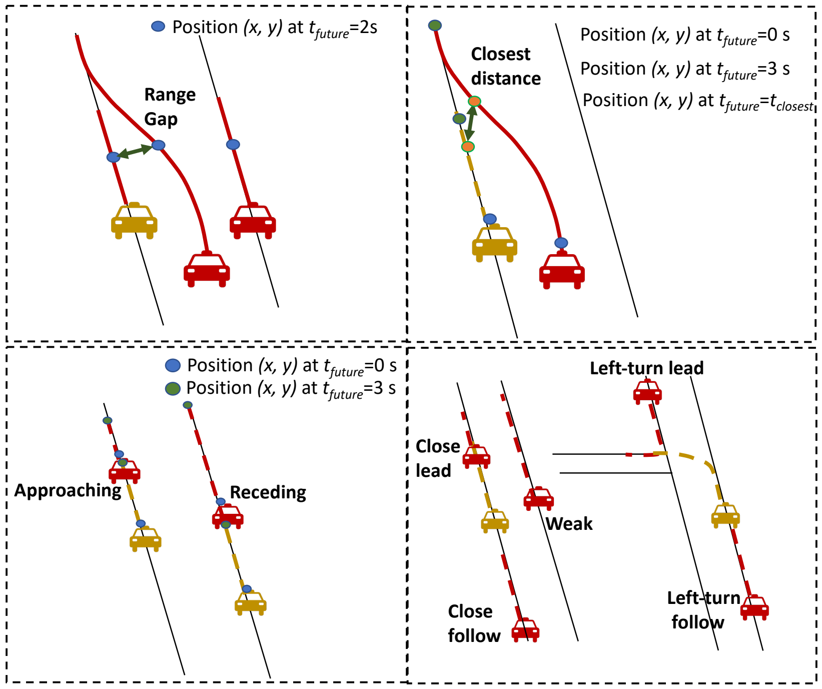

Position (x, y) at t_future=2s
Range Gap
Closest distance
Position (x, y) at t_future=0 s
Position (x, y) at t_future=3 s
Position (x, y) at t_future=t_closest
Position (x, y) at t_future=0 s
Position (x, y) at t_future=3 s
Approaching
Receding
Left-turn lead
Close lead
Weak
Close follow
Left-turn follow

Figure 6.4: Our proposed interaction-based pretext tasks, range-gap prediction(top-left), closest-distance prediction(top-right), direction of movement prediction(bottom-left) and type of interaction prediction(bottom-right), as described in Sec. 6.3.3.

with respect to the target agents as obtained from Sec. 6.3.2.

$$\boldsymbol{\mathcal{L}}_{\text{CD}}(\boldsymbol{\Theta}_{\text{CD}}) = \frac{1}{k_{\text{target}}} \sum_{j=1}^{k_{\text{target}}} L_{\text{cross-entropy}}(d_j^{\text{CD}}, y_j^{\text{CD}}) \tag{6.11}$$

Benefit to final forecasting performance: Estimating the closest distance between agents enables the model to better capture spatial relationships and constraints between them. The model can identify potential collisions and navigate around them. This is valuable in constrained environments where agents must avoid collisions while planning their trajectories. Predicting the closest distance between agents can also help the model understand interactions such as yielding or merging, leading to more accurate forecasts in these complex scenarios.

C. Direction of Movement Prediction

In this pretext task, the main idea is to learn to predict whether two agents are moving closer or receding from each other based on their past trajectory information. This pretext task focuses on understanding the relative motion between agents and determining the nature of their interaction. By predicting the direction of movement between agents, the model can learn to understand the influence agents exert on each other and gain insights into potential collisions or coordinated movements.

Formulation: Let the pretext task feature extractor for this task to be denoted by f_{DM} with the parameters $\boldsymbol{\Theta}_{\text{DM}}$. We consider the ground-truth direction of movement between two agents i, which is the target agent, and j to be calculated by the difference in intial and final distances between them. $D_j^{\text{init}} = ||p_{\text{target}}^{t=1} - p_j^{t=1}||$, and $D_j^{\text{final}} = ||p_{\text{target}}^{t=T_c} - p_j^{t=T_c}||$. The direction of movement can be determined by calculating the difference between the initial and final distances as: $\text{Dir}_j^{\text{GT}} = D_j^{\text{final}} - D_j^{\text{init}}$. If $\text{Dir}_j^{\text{GT}} > 0$, then agents are receding. If $\text{Dir}_j^{\text{GT}} < 0$, the agents are moving closer. We want to setup this task as a classification task, and choose three classes with labels y_j^{DM}:

$$y_j^{\text{DM}} = \begin{cases} 0 & \text{if} \quad \text{Dir}_j^{\text{GT}} \geq 2 \\ 1 & \text{if} \quad \text{Dir}_j^{\text{GT}} \leq -2 \\ 2 & \text{otherwise} \end{cases} \tag{6.12}$$

The auxiliary direction of motion feature predictor takes the following inputs: the agent-to-agent interaction feature from the standard motion forecasting architecture as described in Sec. 6.2.2 represented by $\tilde{\boldsymbol{v}}_{\text{target}}$ and $\tilde{\boldsymbol{v}}_j$, and the distance between them at $t_p = 0$ given by $d^0_{\text{target}:j}$. The output is a predicted direction of movement between the target agent and j.

$$d_j^{\text{DM}} = \text{Softmax}(f_{\text{DM}}(\tilde{\boldsymbol{v}}_{\text{target}} - \tilde{\boldsymbol{v}}_j, d^0_{\text{target}:j})) \tag{6.13}$$

The objective function $\boldsymbol{\mathcal{L}}_{\text{DM}}$ for guiding the direction of motion prediction is formulated using cross-entropy loss as shown in Eq. (6.14). k_{target} is the number of interacting agents with respect to the target agents as obtained from Sec. 6.3.2.

$$\boldsymbol{\mathcal{L}}_{\text{DM}}(\boldsymbol{\Theta}_{\text{DM}}) = \frac{1}{k_{\text{target}}} \sum_{j=1}^{k_{\text{target}}} L_{\text{cross-entropy}}(d_j^{\text{DM}}, y_j^{\text{DM}}) \tag{6.14}$$

Benefit to final forecasting performance: By predicting the direction of movement between agents, the model learns to represent and capture the relative motion between them more effectively. If the agents are predicted to move closer, the model can anticipate a possible collision and adjust the motion forecasts to avoid it. Predicting the direction of movement can help the model identify coordinated movements between agents, such as merging. This is also useful in scenarios when vehicles are accelerating or decelerating, where ego can adjust their trajectories to move safely. In a multi-lane traffic scenario, vehicles may need to change lanes to navigate around slower vehicles or prepare for an upcoming exit. Predicting the direction of movement between vehicles in adjacent lanes can help the model identify when a vehicle is actively changing lanes or maintaining its position.

D. Type of Interaction Prediction

In this pretext task, the main idea is to enhance interaction modeling for trajectory prediction by classifying the types of interactions between agents. This is a classification task that involves predicting among five interaction types {'close-lead', 'close-follow', 'left-turn-lead', 'left-turn-follow', 'weak'}, based on the past trajectory information of agents. By classifying the type of interaction, the model can better capture the nuanced dynamics of agent behavior, cooperation and collision, which can lead to more accurate motion forecasts.

Formulation: Let the pretext task feature extractor for this task to be denoted by f_{TI} with the parameters $\mathbf{\Theta}_{\text{TI}}$. We first describe how we create the pseudo ground-truth labels for this setup, influenced by M2I [157]. We consider the type of interaction between two agents i, which is the target agent, and j. We want to setup this task as a classification task with the pseudo-label for agent j represented as y_j^{TI}. We first check the target agent intention (obtained from Sec. 6.3.2. If the target agent intention is straight, we classify the j agents not in the same lane as 'weak'. If the target agent intention is 'left-turn' or 'left-turn-waiting', we label the agents j in the right neighboring lane as 'weak'. If the target agent intention is 'right-turn' or 'right-turn-waiting', we label the agents j in the left neighboring lane as 'weak'.

For the strong interactions, we first compute the closest spatial distance the target agent and agent j:

$$d_I = \min_{t_1=1}^{T_c} \min_{t_2=1}^{T_c} ||p_{\text{target}}^{t_1} - p_j^{t_2}|| \tag{6.15}$$

If $d_I > \epsilon_d$, where ϵ is a user defined threshold, the agents never get close to each other and thus we label the relation y_j^{TI} as 'weak'. Otherwise, we obtain the time step from each agent at which they reach the closest spatial distance, such that:

$$t_1 = \arg \min_{t_1=1}^{T_c} \min_{t_2=1}^{T_c} ||p_{\text{target}}^{t_1} - p_j^{t_2}|| \tag{6.16}$$

$$t_2 = \arg \min_{t_2=1}^{T_c} \min_{t_1=1}^{T_c} ||p_{\text{target}}^{t_1} - p_j^{t_2}|| \tag{6.17}$$

When $t1 > t2$, we define that target agent *follows* agent j, as it takes longer for target agent to reach the interaction point. The label y_j^{TI} for agent j is 'left-turn-lead' if the target intention is {'left', 'left-turn-waiting'}, and 'close-lead' otherwise. Similarly, if $t1 < t2$, we define that target agent *leads* agent j. The label y_j^{TI} for agent j is 'left-turn-follow' if the target intention is {'left', 'left-turn-waiting'}, and 'close-follow' otherwise.

The auxiliary direction of motion feature predictor takes the following inputs: the agent-to-agent interaction feature from the standard motion forecasting architecture as described in Sec. 6.2.2 represented by $\tilde{\boldsymbol{v}}_{\text{target}}$ and $\tilde{\boldsymbol{v}}_j$, and the distance between them at $t_p = 0$ given by $d^0_{\text{target}:j}$. The output is a predicted type of interaction relation between the target agent and j.

$$d_j^{\text{TI}} = \text{Softmax}(f_{\text{TI}}(\tilde{\boldsymbol{v}}_{\text{target}} - \tilde{\boldsymbol{v}}_j, d^0_{\text{target}:j})) \tag{6.18}$$

The objective function $\boldsymbol{\mathcal{L}}_{\text{TI}}$ for guiding the type of interaction relation prediction is formulated using cross-entropy loss as shown in Eq. (6.19). k_{target} is the number of interacting agents with respect to the target agents as obtained from Sec. 6.3.2.

$$\boldsymbol{\mathcal{L}}_{\text{TI}}(\boldsymbol{\Theta}_{\text{TI}}) = \frac{1}{k_{\text{target}}} \sum_{j=1}^{k_{\text{target}}} L_{\text{cross-entropy}}(d_j^{\text{TI}}, y_j^{\text{TI}}) \tag{6.19}$$

Benefit to final forecasting performance: Knowing the type of interaction can aid the model in making more informed decisions when predicting trajectories. For example, if the model classifies an interaction as 'close-follow', it can adjust the following agent's trajectory to maintain a safe distance from the leading agent. At intersections, vehicles may engage in different types of interactions, such as taking a left turn in front of another vehicle or following closely behind a vehicle making a left turn. In multi-lane traffic scenarios, vehicles may engage in different types of interactions, such as leading or following another vehicle during a lane change or overtaking maneuver. By predicting the type of interaction (e.g., 'close-lead', 'close-follow'), the model can better understand the dynamics of lane changing and overtaking. This leads to more accurate motion forecasts that account for the specific interaction types.

6.3.4 Training Scheme

We train both the pretext losses and the main task in a simultaneous manner, as described in equation Eq. (6.4). A crucial aspect of our approach is the design of loss functions that specifically aid in the modeling of interactions between agents. To achieve this objective, we restrict the propagation of pretext loss gradients solely to the agent-to-agent interaction module. As a result, the gradients only affect the function f_{interact}, as defined in equation Eq. (6.3). In practice, this can be accomplished through the use of a stop-gradient operation. We only back-propagate the loss through the target agent of interest and the other agents that interact with it, given a particular scene. Furthermore, since we predict K modes, as described in Sec. 6.2, the pretext task predictions for interactions must reflect this. Therefore, we predict K modes for each of the following functions: f_{RG}, f_{CD}, f_{DM}, and f_{TI}. For each target agent, we select the mode that produces the minimum loss with respect to the main forecasting task.

6.4 Experiments

To train and evaluate the model using pretext tasks, we first introduce the dataset used. Then we describe the models used to evaluate forecasting performance, and illustrate the situations where pretext tasks can enhance predictions.

6.4.1 Dataset

The Argoverse v1.1 [26] platform offers a comprehensive dataset designed for the training and evaluation of models. The main objective here is the prediction of 3 seconds of future movements, leveraging 2 seconds of past observations. There is a clear division between training and validation sets, avoiding any geographical overlap. In particular, each sequence presents the positions of every actor in a scene, annotated at a frequency of 10Hz, for the past 2 seconds. Each sequence identifies an 'agent' of interest whose future 3-second movement is considered for assessment. Both training and validation sets also indicate the future positions of all actors within a 3-second time frame, labeled at 10Hz. The dataset provides HD map information for all sequences labeled in Miami and Pittsburgh.

The original training set consists of 205,942 sequences, while the validation set has 39,472 sequences. We curate and augment the Argoverse v1.1 using Algorithm 2 with interaction information and pretext task pseudo labels, as described in Sec. 6.3.2 and

Sec. 6.3.3. In terms of data statistics, this results in 93,200 unique training sequences and 18592 unique validation sequences. In terms of size, the Waymo Interactive Split [38] and our dataset are comparable. While Waymo Interactive Split assesses on 20,000 scenes with 2 agents, our validation set tests 18,502 scenes, encompassing 47,400 agents. Comparatively, the Waymo Interactive Split, features only two interacting agents per scene, whereas our samples may contain two or more interacting agents.

6.4.2 Metrics

We follow the Argoverse [26] benchmark and use the following metrics for evaluation: minimum final displacement error(min-FDE), and miss rate (MR). Where the choice of K is required, to define the top K trajectories to be used for evaluation of a metric (for example min-FDE (K = k) on Argoverse), we use k=6.

We also propose three additional metrics to assess the quality of interaction prediction. The first one, termed as interactive min-FDE (i-min-FDE), is calculated for all agents interacting with the agent of interest. The i-min-FDE is examined in two scenarios: one including all interacting agents, and the other involving only strongly interacting agents, explicitly excluding any weak interactions as delineated in Sec. 6.3.3.
The second one is termed as non-interactive-min-FDE (ni-min-FDE) which is calculated for all non-interactive sequences. The motivation here is that since pretext tasks are proposed to improve interaction modeling, non-interactive sequences should have similar performance as baseline.
The third metric called Collision Avoidance Measure (CAM) is designed to evaluate the performance of predictive models in terms of their ability to forecast and avoid potential collision scenarios between multiple agents in a given environment. The CAM is calculated by iterating over all time steps and pairwise combinations of agents. For each pair of agents i and j at time t, we calculate the Euclidean distance between their predicted positions and their actual positions. If predicted distance is less than a distance-threshold (indicating a predicted collision or near-collision scenario) and ground-truth is greater than or equal to the distance-threshold (indicating that no collision or near-collision actually occurred), we increment a counter. The final value of CAM is the total number of these predicted collisions or near-collision scenarios across all time steps and pairs of agents, divided by the total number of scenes. CAM $= \frac{1}{s_{\text{total}}} \sum_{t=1}^{T} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mathcal{I}\{||\mathcal{Y}_{i,t} - \mathcal{Y}_{j,t}|| < d_{\text{CAM}} \cap ||\mathcal{Y}_{i,t}^{GT} - \mathcal{Y}_{j,t}^{GT}|| >= d_{\text{CAM}}\}$. Here, the function $\mathcal{I}(.)$ is an indicator function that equals 1 if the condition in parentheses is true and 0 otherwise, and s_{total} is the total number of evaluation sequences. Thus, a lower CAM indicates a better predictive model in terms of its ability to avoid potential collisions.

6.4.3 Implementation Details

We discuss data pre-processing, training details and base model architecture in this section.

Data pre-processing: In order to achieve normalize data, the coordinate system corresponding to each sequence is subjected to translation and rotation. The objective is to reposition the origin at the present position of the acting agent (defined as 'target agent') when $t = 0$, while aligning the positive x-axis with the agent's current direction. This direction is determined based on the orientation of the agent between the location at $t = -1$ and the location at $t = 0$.

Training: The model input includes all actors and lanes that interact with the target agent, including the agent itself. The model's training is executed on four TITAN-X GPUs with a batch size of 32. The optimizer used was Adam, with an initial learning rate of 1×10^{-3}. The rate was decayed to 1×10^{-4} at 93k steps, taking approximately five hours in total to finish. λ in Eq. (6.4) is set to 1.

Base Model Architecture: For the agent feature extractor, the architecture is similar to Lane-GCN [88]. We use a 1D CNN to process the trajectory input. The output is a temporal feature map, whose element at $t = 0$ is used as the agent feature. The network has three groups/scales of 1D convolutions. Each group consists of two residual blocks, with the stride of the first block as 2. Feature Pyramid Network (FPN) fuses the multi-scale features, and applies another residual block to obtain the output tensor. For all layers, the convolution kernel size is 3 and the number of output channels is 128. Layer normalization and Rectified Linear Unit (ReLU) are used after each convolution. The map feature extractor has two LaneConv residual blocks which are the stack of a LaneConv(1) and a linear layer, as well as a shortcut. All layers have 128 feature channels. Layer normalization and ReLU are used after each LaneConv and linear layer.
For the map-aware agent feature (M2A) module, the distance threshold is 12m. The one M2A interaction module has two residual blocks, which consist of a stack of an attention layer and a linear layer, as well as a residual connection. The A2A layer has four such blocks. All layers have 128 output feature channels. The pretext task encoder that takes as input the A2A vector has two residual blocks, which also have 128 dimensional output. Taking the A2A actor features as input, our trajectory decoder is a multi-modal prediction header that outputs the final motion forecasting. For each agent, it predicts $K = 6$ possible future trajectories and confidence scores. The header has two branches, a regression branch to predict the trajectory of each mode and a classification branch to predict the confidence score of each mode.

Model	A2A	Pretext Task	$\text{minFDE}_1 \downarrow$	$\text{minFDE}_6 \downarrow$	$\text{MR}_6 \downarrow$	$\text{Improvement}_{\text{minFDE}_6} \uparrow$
Without A2A	✗	✗	4.006	1.516	19.879	-
Without A2A With Pretext	✗	✓(All)	3.999	1.512	19.441	-
Baseline	✓	✗	3.814	1.351	16.686	-
Ours	✓	Range-gap	3.253	1.305	15.382	+3.4%
Ours	✓	Closest-distance	**3.230**	1.295	15.209	+4.1%
Ours	✓	Direction of Movement	3.284	**1.279**	**14.866**	+5.3%
Ours	✓	Type Of Interaction	3.340	1.306	15.539	+3.3%

Table 6.1: Motion forecasting performance for target agent on Argoverse v1.1 [26] validation set curated with only interactive examples via Algorithm 2, with and without our proposed pretext tasks (see Sec. 6.3.3)

6.4.4 Baselines and Proposed Model

We train different versions of our base model with and without the interaction component (A2A layer) and the pretext tasks. We compare the performance of these models on our validation set. We have the following four variants:

- Without A2A: Train the base model described in Sec. 6.4.3 with the A2A layer, but without the pretext tasks.
- Without A2A With Pretext: Train the base model without the A2A layer, but with the pretext task losses added. The A2A layer has a stop gradient for the pretext loss as described in Sec. 6.3.4.
- Baseline: Train the base model without the A2A layer and without the pretext tasks.
- Proposed Model: Train the baseline model with both the A2A layer and the proposed pretext tasks. This model also has a stop gradient operation so that the pretext loss only affects the A2A layer.

6.5 Results

In this section, we describe our ablation studies and quantitative and qualitative results.

Ablation: Our findings reveal several distinct trends, as displayed in Tab. 6.1. This table outlines the metrics of the target agent, as defined by Argoverse, on our specially-selected, interaction-dense validation dataset.

We observe that the model lacking the A2A layer (shown in the first row) underperforms our established baseline (shown in the third row) by a notable 11%. This underscores the considerable influence of the A2A layer when assessing performance on our interaction-heavy dataset.

The second model, which is without the A2A layer but incorporates pretext tasks (calculated as the cumulative loss from all proposed pretext tasks), mirrors the performance of the first model closely. This suggests that the pretext tasks do not impact the M2A layer, the map, or the agent encoder, due to the application of the stop gradient operation. Instead, they only influence the A2A layer. This finding is pivotal in validating our proposition that the pretext tasks assist in interaction modeling.

Comparison with state-of-the-art: Across all proposed pretext tasks, we also noticed a significant improvement in the model's performance w.r.t state-of-the-art baseline — ranging from 3.3% to 5.3% across the minFDE$_6$ metric. We centered our attention on min-FDE in this table because of its importance in making precise long-term predictions, which are vital for safety. The most effective predictors for this dataset were direction of movement and closest distance. The range-gap prediction and the type of interaction prediction under-perform the above mentioned tasks. In examining the reasons for this disparity, we hypothesize that the structure of the range-gap prediction task, which is a regression task, could have contributed to its lesser performance. Regression tasks are typically more challenging to learn compared to classification tasks due to their demand for more complex learning mechanisms. Furthermore, the task of type of interaction prediction presented two significant difficulties. Firstly, the task was heavily imbalanced, and secondly, it did not offer precise information about the trajectory endpoints. The lack of fine-grained information in this task could have inhibited the model's ability to make accurate long-term predictions. These observations underscore the need to carefully select and design pretext tasks when building models for interaction-heavy datasets.

Evaluation on Proposed Metrics: A consistent pattern can be observed in Tab. 6.2, where we evaluate our proposed metrics for both the baseline and the suggested pretext tasks. We assess the performance using the metrics i-minFDE$_6$ and CAM$_6$, not only for the target agent but also for all interacting agents. This comprehensive evaluation is crucial as it is not only the predictions for the primary agent that matter, but also those for the surrounding agents in an interactive scenario. Our findings indicate that there are more substantial improvements (upto 8.1% for all interacting agents, 7.9% for only strongly interacting agents, 8.6% for collision awareness) over the baseline when using these proposed metrics. This suggests that the final positions of the interacting agents are predicted more accurately than with the baseline, and there's improved collision awareness. These metrics provide a more detailed understanding of the model's performance in predicting interac-

Model	**A2A**	**Pretext Task**	**i-minFDE$_6$ ↓** All-Interactive	Strong-Interactive	**ni-min-FDE$_6$ ↓**	**CAM$_6$ ↓**
Baseline	✓	✗	1.279	1.320	1.459	1.472
Ours	✓	Range-gap	1.202 (+6.0%)	1.242 (+5.9%)	1.468	1.394 (+5.3%)
Ours	✓	Closest-distance	1.192 (+6.8%)	1.235 (+6.4%)	1.446	**1.346** (+8.6%)
Ours	✓	Direction of Movement	**1.175** (+8.1%)	**1.216** (+7.9%)	1.441	1.375 (+6.6%)
Ours	✓	Type Of Interaction	1.183 (+7.5%)	1.226 (+7.1%)	1.455	1.376 (+6.5%)

Table 6.2: Motion forecasting performance for interactive agents on Argoverse v1.1 [26] validation set curated with only interactive examples via Algorithm 2, with our proposed pretext metrics (see Sec. 6.4.2)

tive scenarios. In addition to the interactive dataset, we maintain a set of non-interactive validation data. Upon evaluating this data using the ni-minFDE$_6$ metric, we observe that the baseline model's performance is nearly on par with the models employing pretext tasks. This observation is interesting — it indicates that the improvements resulting from the inclusion of pretext losses are primarily manifested in datasets with heavy interaction. Specifically, the pretext tasks do not notably enhance performance on non-interactive data. This finding aligns with our objective to demonstrate that pretext tasks are particularly beneficial in modeling interactions.

Qualitative Results: Illustrations of improvements provided by the pretext task models are shown in Fig. 6.5. Each illustration portrays a unique urban driving scenario, and details how the introduction of the pretext task model enhances the predictive capabilities. In these diagrams, we compare the ground-truth with the baseline model's prediction, the model's prediction when interactions with other agents have been disconnected, and the proposed model's prediction. These illustrations serve to underscore the effectiveness of our pretext task models in a real-world context.

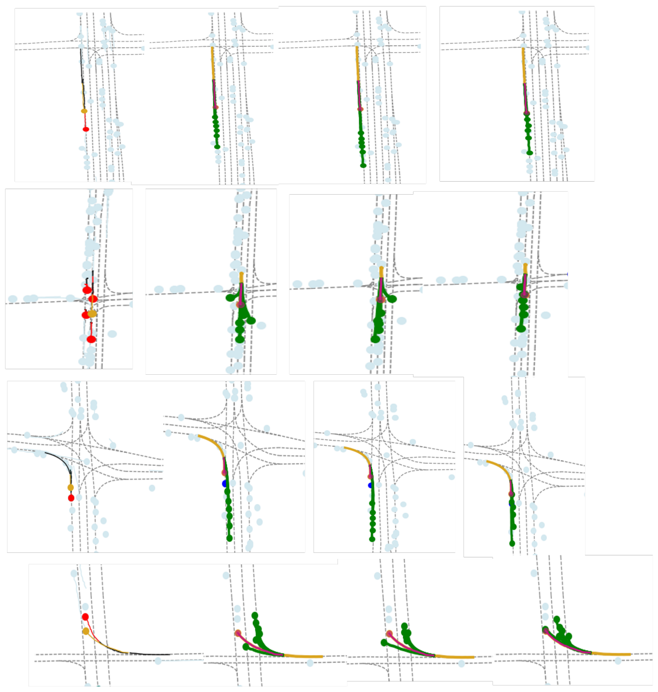


Figure 6.5: Motion forecasting on curated, interaction-heavy Argoverse [26] validation. We present four challenging scenarios for analysis. The first column depicts the scene featuring the target agent and the curated interactive agents. The second column contains predictions made by the baseline model [88]. The third column displays the predictions of our proposed model when the connections to the interactive agents are disconnected. The fourth and final column features predictions from our model when regularized with the pretext loss. The baseline model fails to accurately forecast any of the scenarios. The first row illustrates a case where the predicted range-gap accurately anticipates at least one future trajectory that evades collision with the forward vehicle. The second row presents a congested situation in which forecasting the closest-distance with interacting agents leads to a future trajectory devoid of collision with the vehicle in front. The third row depicts a similar situation, but in the context of predicting direction of movement - both the baseline and the model without interacting agent information result in a collision course with the agent ahead, whereas our model proposes a trajectory closely aligned with the ground truth, avoiding collision.The final row presents a turning scenario wherein the type of interaction is predicted. Our proposed model forecasts a 'close-follow' situation with the interacting vehicle and identifies a potential future trajectory that is most closely aligned with the ground truth, while the baseline and the model without knowledge of interacting agents fail in terms of the MR_6 metric. Please refer to Sec. 6.3.3 for details.

6.6 Chapter Conclusion

In this work, we improve motion forecasting with a particular focus on interactive scenarios. Our first contribution is a simple yet effective approach for interaction modeling. We introduce four interaction-specific pretext tasks, which incorporate domain knowledge and are learned concurrently with the main task. This methodology enables the pretext task to effectively capture the dependencies between agents' movements. Our second contribution is a novel method for curating datasets, which allows for the explicit labeling of interacting pairs. This approach is crucial for the training of interaction-specific pretext tasks and the generation of necessary pseudo-labels. The third contribution is the development of new metrics designed to offer a more accurate evaluation of performance in scenarios characterized by a high degree of interaction. These metrics provide a detailed and nuanced understanding of the model's performance. Finally, we demonstrate that our proposed methodology significantly outperforms the baseline in both quantitative measurements and qualitative assessments. In summary, our work advances the understanding and application of interaction modeling in motion forecasting, providing useful tools and methods that can be further built upon in future research.

Part III

Conclusions and Future Work

Chapter 7

Conclusion

This dissertation presents multiple contributions related to perception and prediction in the context of autonomous driving, with an emphasis on improving context modeling using attention mechanisms and self-supervised learning. We close with a summary of our novel contributions followed by a discussion of challenges and directions for future work in this area.

7.1 Summary of Contributions

7.1.1 FANTrack: 3D Multi-Object Tracking with Feature Association Network (Chapter 3)

We presented FANTrack, an advanced 3D object tracking framework that significantly enhances the data association phase inherent to multi-object tracking tasks. This new formulation exhibits potential to mitigate common challenges associated with correctly associating detections across time and space, thus fostering advancements in the domain of dynamic scene understanding in autonomous driving. Our contributions can be summarized as follows:

- Our approach can solve the multi-target association problem by performing inference using CNNs.

- It can integrate image based appearance and 3D bounding box features to get a discriminative as well as generalized feature representation, thereby learning a robust cost function for association.
- We show competitive qualitative and quantitative 3D tracking results compared to the state of the art on the KITTI tracking dataset.

7.1.2 SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection (Chapter 4)

We proposed SA-Det3D, which includes self-attention variants for 3D object detection that augment convolutional features, model long-range interactions and can sample representative features for scalability and distinct feature descriptor generation. Our contributions can be summarized as follows:

- We propose a generic globally-adaptive context aggregation module that can be applied across a range of modern architectures including BEV, voxel, point and point-voxel based 3D detectors. We show that we can outperform strong baseline implementations by up to 1.5 3D AP (average precision) while simultaneously reducing parameter and compute cost by 15-80% and 30-50%, respectively, on the KITTI validation set.
- We design a scalable self-attention variant that learns to deform randomly sampled locations to cover the most representative and informative parts and aggregate context on this subset. This allows us to aggregate global context in large-scale point-clouds like nuScenes and Waymo Open dataset.
- Extensive experiments demonstrate the benefits of our proposed FSA/DSA modules by consistently improving the performance of state-of-the-art detectors on KITTI, nuScenes and Waymo Open dataset.

7.1.3 SSL-Lanes: Self-Supervised Learning for Motion Forecasting in Autonomous Driving (Chapter 5)

We proposed SSL-Lanes, an approach that extends self-supervised learning (SSL) principles to motion forecasting, presenting the first systematic study on incorporating self-supervision in a standard data-driven motion forecasting model. Our contributions can be summarized as follows:

- We demonstrate the effectiveness of incorporating self-supervised learning in motion forecasting. Since this does not add extra parameters or compute during inference, SSL-Lanes achieves the best accuracy-simplicity-efficiency trade-off on the challenging large-scale Argoverse benchmark.

- We propose four self-supervised tasks based on the nature of the motion forecasting problem. The key idea is to leverage easily accessible map/agent-level information to define domain-specific pretext tasks that encourage the standard model to capture more superior and generalizable representations for forecasting, in comparison to pure supervised learning.

- We further design experiments to explore why forecasting benefits from SSL. We provide extensive results to hypothesize that SSL-Lanes learns richer features from the SSL training as compared to a model trained with vanilla supervised learning.

7.1.4 SSL-Interactions: Pretext Tasks for Interactive Trajectory Prediction(Chapter 6)

We proposed SSL-Interactions, which improves motion forecasting by utilizing pretext tasks to capture interaction dynamics and regularize the model by incorporating interaction-specific knowledge. Our contributions can be summarized as follows:

- We propose a framework called SSL-Interactions, that leverages pretext tasks to improve interaction modeling for motion forecasting. Specifically, we develop four pretext tasks, designed to capture various aspects of interaction based on domain-specific knowledge.

- The majority of recent motion forecasting datasets do not clearly identify pairs of interacting agents, relying instead on the implicit interaction modeling conducted via end-to-end training. We propose a simple but effective way to curate interaction-specific scenarios from datasets and to explicitly label pairs of interacting agents within a given scenario. This approach is crucial for generating pseudo-labels for interaction-centric pretext tasks.

- Through empirical evaluation, we demonstrate that our proposed framework can surpass a state-of-the-art motion forecasting method both quantitatively, with up to an 8% improvement, and qualitatively. Furthermore, we introduce three new metrics specifically designed to evaluate predictions within interactive scenes.

7.2 Future Work

We end this dissertation with potential future work for perception and prediction with application to autonomous driving.

7.2.1 End-to-End Learning for 3D Object Detection and Data-Association

While FANTrack in Chapter 3 has proven effective with its two-step process - 3D object detection followed by data-association - this methodology inherently treats these tasks as distinct stages. The learning from the data association stage do not influence the object detection stage, thereby potentially omitting valuable information. An end-to-end trainable framework could mitigate this issue, as back-propagation during training would fine-tune both the 3D object detection and data-association sub-components simultaneously. This could lead to an object detection process that is intrinsically optimized to improve data association, potentially producing more distinctive detections that enable more reliable object identification across different frames. In addition to refining the training process, an end-to-end framework could allow the data association process to consider information about the uncertainty or quality of individual detections when determining associations. A preliminary approach is presented by PnP [90], which can be extended. Another valuable future direction might involve balancing the improved performance offered by end-to-end learning with maintaining a model's interpretability, to ensure alignment with human reasoning in decision-making processes.

One of the other crucial avenues for improving FANTrack involves robustness to occlusions and low-illumination conditions. These are scenarios where an object is either temporarily obscured by another or lacks clear visibility due to poor lighting. Utilizing generative models such as diffusion models [58] could bring novel solutions to problems of occlusion and low-illumination in 3D object tracking. For handling occlusions, one way of employing diffusion models could be to generate probable trajectories of the occluded objects. Training the diffusion model on a large dataset of object trajectories could enable it to generate plausible predictions during occlusions that follow the physical laws of motion. When addressing low-illumination scenarios, generative models can be used for image enhancement, essentially reconstructing a well-lit version of a low-light scene. For instance, diffusion models, trained on a large dataset of low-light and corresponding normal-light images, could be tasked with transforming a low-light image into its normal-light version. It would be essential to ensure that the computational cost of the diffusion model does not slow down the tracking process too much, which could limit the real-world applicability of the system. Future work could explore efficient ways of integrating these generative models into the tracking system to maintain a balance between performance improvement and computational efficiency.

7.2.2 Representation Learning for 3D Object Detection Using Unlabeled Data

A major challenge in 3D object detection, as presented in Chapter 4 is the limited availability of annotated data, largely due to the labor-intensive nature of the labeling process. By designing methods that can learn meaningful representations from unlabeled data, or that can effectively transfer knowledge from labeled to unlabeled data, the reliance on extensively annotated datasets could be significantly reduced.

Segment Anything (SAM [76]) recently introduces a novel "foundation model" approach for image segmentation, drawing inspiration from natural language processing (NLP). The core idea involves training a model that can segment images based on a range of prompts, much like a language model predicting the next token given prior context. Prompts can vary from foreground/background points, rough box or mask annotations, to free-form text. The model is trained to always predict a valid segmentation mask, even when the prompt is ambiguous. This capability enables the model to generalize to a variety of downstream segmentation tasks, simply by feeding in different prompts. The training algorithm is based on simulating a sequence of prompts for each training sample and comparing model predictions against ground truth.

When extended to 3D object detection, the foundation model approach could address several limitations of current state-of-the-art models. Similar to the segmentation task, a model could be trained to perform 3D object detection given a range of prompts, improving the model's flexibility and generalization capability. Training the model to produce valid object detections even in the presence of ambiguous prompts could enhance its robustness to varying real-world conditions. Furthermore, the use of prompts could facilitate zero-shot transfer, reducing the need for extensive task-specific training. Lastly, this approach could provide a novel way to leverage unlabeled data, by training the model to make sensible predictions even when only partial or ambiguous annotations are available.

7.2.3 Effective Pretext Tasks for Motion Forecasting

One potential limitation of our SSL-Lanes in Chapter 5 and SSL-Interactions in Chapter 6 framework for motion forecasting lies in the fact that we employ different losses for our formulation at a fixed 1:1 ratio without optimizing them. Additionally, we only leverage one pretext task at a time, not exploring the synergy that might come from combining different tasks. A promising area for future work would be to incorporate meta-learning approaches to determine an optimal mix of pretext tasks and automatically adjust their balance [68], which we anticipate could further boost forecasting performance.

Another aspect to consider is that in our SSL-Lanes framework, we observe improvements with SSL-pretext tasks predominantly in scenarios that do not involve multiple interacting agents. Conversely, for SSL-Interactions, we specifically construct pretext tasks to model interaction. Going forward, it would be intriguing to investigate how to merge geometric and interaction-based pretext tasks, considering both road features and agent interactions. We are particularly interested in examining this within the context of the interaction split of the Waymo Open Motion dataset [38].

Lastly, in terms of generalization, our exploration in SSL-Lanes only addresses implicit data imbalance in comparison to strictly supervised training on the same dataset from which training samples are drawn. We are keen to examine the capacity of both SSL-Lanes and SSL-Interactions to generalize to other datasets without the need for re-training.

7.2.4 Interaction Modeling for Motion Forecasting and Large Language Models

The work presented in [81] is one of the first works that proposed to co-train a sentence generator with an attention-based trajectory predictor, generating sentences that hypothesize the description of the full trajectory, given the past. Comparing this work with Chapter 6, predicting the linguistic description for a trajectory can also be viewed as a regularizing pretext task. However, the focus of the paper leans more towards establishing an interpretability mechanism to ensure autonomous vehicles make decisions based on valid reasoning, not merely achieving benchmark results. The methodology does not leverage large language models and mainly generates straightforward descriptions of trajectories. Thus, the realm of interaction modeling for trajectory prediction using large language models remains largely unexplored.

A more recent study, [199], takes initial steps in examining the potential of large language models (LLMs) for modeling human behavior to enhance human-robot interaction

(HRI). Given the complexities associated with accurately modeling human behavior, which often requires extensive prior knowledge or substantial interaction data, the authors suggest employing LLMs. These models, trained on copious amounts of human-generated text data, could function as zero-shot human models in HRI contexts.

Therefore, it would be intriguing to investigate further the zero-shot modeling capabilities of LLMs. Specifically, it would be beneficial to determine whether these capabilities could be effectively transferred to the task of interaction modeling for motion forecasting within the context of autonomous driving.

References

[1] Tracking multiple targets based on min-cost network flows with detection in rgb-d data. *Int. J. Comput. Sci. Eng.*, 15(3-4):330–339, January 2017.

[2] Elmira Amirloo Abolfathi, Mohsen Rohani, Ershad Banijamali, Jun Luo, and Pascal Poupart. Self-supervised simultaneous multi-step prediction of road dynamics and cost map. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 8494–8503. Computer Vision Foundation / IEEE, 2021.

[3] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 961–971. IEEE Computer Society, 2016.

[4] Anton Andriyenko, Konrad Schindler, and Stefan Roth. Discrete-continuous optimization for multi-target tracking. In *CVPR*, 2012.

[5] Donald B. Reid. "an algorithm for tracking multiple targets". volume 24, pages 1202 – 1211, 02 1978.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization, 2016.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[8] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

[9] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium, IV 2019, Paris, France, June 9-12, 2019*, pages 1426–1433. IEEE, 2019.

[10] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[11] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia, and Arturo De La Escalera. BirdNet: a 3d object detection framework from LiDAR information. In *ITSC*. IEEE, 2018.

[12] Jérôme Berclaz, François Fleuret, Engin Türetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1806–1819, 2011.

[13] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

[14] Prarthana Bhattacharyya and Krzysztof Czarnecki. Deformable PV-RCNN: improving 3d object detection with learned deformations. *CoRR*, abs/2008.08766, 2020.

[15] Prarthana Bhattacharyya, Chengjie Huang, and Krzysztof Czarnecki. Sa-det3d: Self-attention based context-aware 3d object detection. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*, pages 3022–3031. IEEE, 2021.

[16] Prarthana Bhattacharyya, Chengjie Huang, and Krzysztof Czarnecki. Ssl-lanes: Self-supervised learning for motion forecasting in autonomous driving. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 1793–1805. PMLR, 2022.

[17] Michael D. Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter, what. *2009 IEEE 12th International Conference on Computer Vision*, pages 1515–1522, 2009.

[18] Asad A. Butt and Robert T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1846–1853, 2013.

[19] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[20] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, volume 11218 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 2018.

[21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9630–9640. IEEE, 2021.

[22] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 9491–9497. IEEE, 2020.

[23] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*, volume 12368 of *Lecture Notes in Computer Science*, pages 624–641. Springer, 2020.

[24] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 947–956. PMLR, 2018.

[25] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual*

Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings, volume 100 of *Proceedings of Machine Learning Research*, pages 86–99. PMLR, 2019.

[26] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8748–8757. Computer Vision Foundation / IEEE, 2019.

[27] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z. Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 389–398. IEEE, 2020.

[28] Qi Chen, Lin Sun, Ernest Cheung, and Alan L. Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[29] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[30] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, Quoc V. Le, Jonathon Shlens, and Dragomir Anguelov. Improving 3d object detection through progressive population based augmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXI*, volume 12366 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2020.

[31] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. 04 2015.

[32] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[33] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[34] Nachiket Deo and Mohan M. Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pages 1179–1184. IEEE, 2018.

[35] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1422–1430. IEEE Computer Society, 2015.

[36] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[37] Thomas E. Fortmann, Yaakov bar shalom, and Molly Scheffe. Multi-target tracking using joint probabilistic data association. volume 2, pages 807 – 812, 01 1981.

[38] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9690–9699. IEEE, 2021.

[39] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan Gehrig. 6d-vision: Fusion of stereo and motion for robust environment perception. volume 3663, pages 216–223, 08 2005.

[40] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3d tracking by detection. *CoRR*, abs/1806.11534, 2018.

[41] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding HD maps and agent dynamics from vectorized representation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11522–11530. Computer Vision Foundation / IEEE, 2020.

[42] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *CoRR*, abs/2006.12671, 2020.

[43] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI Dataset. In *IJRR*, pages 1229–1235, 2013.

[44] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[45] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. HOME: heatmap output for future motion estimation. In *24th IEEE International Intelligent Transportation Systems Conference, ITSC 2021, Indianapolis, IN, USA, September 19-22, 2021*, pages 500–507. IEEE, 2021.

[46] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *ICML*, 2017.

[47] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira E. Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *10th International Conference on Learning Representations, ICLR 2022, Conference Track Proceedings*, 2022.

[48] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019.

[49] Neil J. Gordon, David Salmond, and A. F. M. Smith. Novel approach to nonlinear / non-gaussian bayesian state estimation. 2004.

[50] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[51] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 15283–15292. IEEE, 2021.

[52] Gültekin Gündüz and Tankut Acarman. A lightweight online multiple object vehicle tracking method. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 427–432. IEEE, 2018.

[53] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: socially acceptable trajectories with generative adversarial networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2255–2264. Computer Vision Foundation / IEEE Computer Society, 2018.

[54] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold Siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[55] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, 2020.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian 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*, pages 770–778. IEEE Computer Society, 2016.

[57] Dirk Helbing and Pé ter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, may 1995.

[58] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[59] Ju Hong Yoon, Chang-Ryeol Lee, Ming-Hsuan Yang, and Kuk-Jin Yoon. Online multi-object tracking via structural constraint event aggregation. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 1392–1400, 2016.

[60] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle trajectory prediction based on motion model and maneuver recognition. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 4363–4369. IEEE, 2013.

[61] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-Excite: Exploiting feature context in convolutional neural networks. In *NeurIPS*, 2018.

[62] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[63] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10998–11006. IEEE, 2020.

[64] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[65] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018.

[66] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. *CoRR*, abs/2109.06446, 2021.

[67] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV) (In Press)*. IEEE, 2019.

[68] Dasol Hwang, Jinyoung Park, Sunyoung Kwon, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J. Kim. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[69] S. Ioffe and C. Szegedyn. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[70] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *CoRR*, abs/2006.10141, 2020.

[71] Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[72] Zia Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1819, Nov 2005.

[73] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *CoRR*, abs/2008.10587, 2020.

[74] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M. Rehg. Multiple hypothesis tracking revisited. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2015.

[75] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arxiv:1412.6980, 2014.

[76] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *CoRR*, abs/2304.02643, 2023.

[77] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR, 2019.

[78] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian D. Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 137–146, 2019.

[79] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. arxiv:1712.02294, 2017.

[80] Harold W. Kuhn. The hungarian method for the assignment problem. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47. Springer, 2010.

[81] Yen-Ling Kuo, Xin Huang, Andrei Barbu, Stephen G. McGill, Boris Katz, John J. Leonard, and Guy Rosman. Trajectory prediction with linguistic representations. In

2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022, pages 2868–2875. IEEE, 2022.

[82] Andrew K. Lampinen, Nicholas A. Roy, Ishita Dasgupta, Stephanie C. Y. Chan, Allison C. Tam, James L. McClelland, Chen Yan, Adam Santoro, Neil C. Rabinowitz, Jane X. Wang, and Felix Hill. Tell me why! explanations support learning relational and causal structure. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 11868–11890. PMLR, 2022.

[83] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[84] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese CNN for robust target association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40, 2016.

[85] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009.

[86] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*. PMLR, 2019.

[87] Yuan Li, Chang Huang, and Ram Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2953–2960. IEEE, 2009.

[88] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pages 541–556. Springer, 2020.

[89] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018.

[90] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11550–11559. Computer Vision Foundation / IEEE, 2020.

[91] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *CoRR*, abs/1805.06334, 2018.

[92] M. Lin, Q. Chen, and S. Yan. Network in network. arxiv:1312.4400, 2013.

[93] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[94] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, pages 2999–3007, 2017.

[95] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 936–944. IEEE Computer Society, 2017.

[96] Todd Litman. Autonomous vehicle implementation predictions: Implications for transport planning. 2015.

[97] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. *CoRR*, abs/2110.05025, 2021.

[98] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 7577–7586. Computer Vision Foundation / IEEE, 2021.

[99] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *CoRR*, abs/2103.00111, 2021.

[100] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. TANet: Robust 3d object detection from point clouds with triple attention. *AAAI*, 2020.

[101] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[102] Haihua Lu, Xuesong Chen, Guiying Zhang, Qiuhao Zhou, Yanbo Ma, and Yong Zhao. SCANet: Spatial-channel attention network for 3d object detection. In *ICASSP*. IEEE, 2019.

[103] Mahendra Mallick and Barbara LA SCALA. Comparison of single-point and two-point difference track initiation algorithms using position measurements. *Acta Automatica Sinica*, 34:258–265, 03 2008.

[104] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *ICCV*, 2019.

[105] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita 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*, pages 9638–9644. IEEE, 2020.

[106] Anton Milan, Seyed Hamid Rezatofighi, Anthony R. Dick, Konrad Schindler, and Ian D. Reid. Online multi-target tracking using recurrent neural networks. *CoRR*, abs/1604.03635, 2016.

[107] Abduallah A. Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian G. Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 14412–14420. Computer Vision Foundation / IEEE, 2020.

[108] Abduallah A. Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian G. Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 14412–14420. Computer Vision Foundation / IEEE, 2020.

[109] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5, 03 1957.

[110] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, Zhifeng Chen, Jonathon Shlens, and Vijay Vasudevan. Starnet: Targeted computation for object detection in point clouds. *CoRR*, abs/1908.11069, 2019.

[111] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Benjamin Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified multi-task model for behavior prediction and planning. *CoRR*, abs/2106.08417, 2021.

[112] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, volume 9910 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2016.

[113] Sang-Il Oh and Hang-Bong Kang. Multiple objects fusion tracker using a matching network for adaptively represented instance pairs. *Sensors (Basel, Switzerland)*, 17, 04 2017.

[114] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention U-Net: Learning where to look for the pancreas, 2018.

[115] Kenji Okuma, Ali Taleghani, Nando de Freitas, James J. Little, and David G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004.

[116] Aljoša Osep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. Combined image-and world-space tracking in traffic scenes. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1988–1995. IEEE, 2017.

[117] Anshul Paigwar, Özgür Erkent, Christian Wolf, and Christian Laugier. Attentional pointnet for 3d-object detection in point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1297–1306. Computer Vision Foundation / IEEE, 2019.

[118] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier. Attentional pointnet for 3d-object detection in point clouds. In *CVPR Workshops*, 2019.

[119] A. O. Pak, Javier Correa, Martin Adams, Daniel Clark, Emmanuel Delande, Jérémie Houssineau, and Jose Franco. Joint target detection and tracking filter for chilbolton advanced meteorological radar data processing. 2016.

[120] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. *CoRR*, abs/2012.11409, 2020.

[121] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[122] Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6024–6033. IEEE Computer Society, 2017.

[123] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2536–2544. IEEE Computer Society, 2016.

[124] Praveena Penmetsa, PJ Sheini, Aibek Musaev, Emmanuel Adanu, and Matthew Hudnall. Effects of the recent autonomous vehicle crashes on public perception of the technology. *IATSS Research*, 45, 06 2021.

[125] Tung Phan-Minh, Elena Corina Grigore, Freddy A. Boulton, Oscar Beijbom, and Eric 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*, pages 14062–14071. Computer Vision Foundation / IEEE, 2020.

[126] Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. *CVPR 2011*, pages 1201–1208, 2011.

[127] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep Hough voting for 3d object detection in point clouds. In *ICCV*, 2019.

[128] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum PointNets for 3d object detection from RGB-D data. In *CVPR*, 2018.

[129] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[130] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[131] Can Qin, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu. PointDAN: A multi-scale 3d domain adaption network for point cloud representation. In *NeurIPS*, 2019.

[132] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, volume 32, 2019.

[133] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2019.

[134] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. *ECCV*, 7573, 01 2012.

[135] H. Ryuhei, F. Aito, N. Keisuke, I. Tomoyuki, and H. Shuhei. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. In *WACV*, pages 1442–1450, 2018.

[136] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *CoRR*, abs/1701.01909, 2017.

[137] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVIII*, volume 12363 of *Lecture Notes in Computer Science*, pages 683–700. Springer, 2020.

[138] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances*

in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 4967–4976, 2017.

[139] Samuel Scheidegger, Joachim Benjaminsson, Emil Rosenberg, Amrit Krishnan, and Karl Granstrom. Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering. *arXiv preprint arXiv:1802.09975*, 2018.

[140] Samuel Schulter, Paul Vernaza, Wongun Choi, and Manmohan Chandraker. Deep network flow for multi-object tracking. *CoRR*, abs/1706.08482, 2017.

[141] Wenling Shang, Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Learning world graphs to accelerate hierarchical reinforcement learning. *CoRR*, abs/1907.00664, 2019.

[142] Sarthak Sharma, Junaid Ahmed Ansari, J. Krishna Murthy, and K. Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. *CoRR*, abs/1802.09298, 2018.

[143] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020.

[144] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[145] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *PAMI*, 2020.

[146] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1708–1716. IEEE, 2020.

[147] Weijing Shi and Raj Rajkumar. Point-GNN: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1711–1719, 2020.

[148] Frank Y. Shih and Christopher C. Pu. A skeletonization algorithm by maxima tracking on euclidean distance transform. *Pattern Recognition*, 28:331–341, 1995.

[149] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. MVX-Net: Multimodal voxelnet for 3d object detection. In *ICRA*. IEEE, 2019.

[150] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay, 2018.

[151] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3786–3795, 2017.

[152] Haoran Song, Di Luan, Wenchao Ding, Michael Yu Wang, and Qifeng Chen. Learning to predict vehicle trajectories with model-based planning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 1035–1045. PMLR, 2021.

[153] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang. VITAL: visual tracking via adversarial learning. arxiv:1804.04273, 2018.

[154] Fan-Yun Sun, Isaac Kauvar, Ruohan Zhang, Jiachen Li, Mykel J. Kochenderfer, Jiajun Wu, and Nick Haber. Interaction modeling with multiplex attention. In *NeurIPS*, 2022.

[155] Lingfeng Sun, Chen Tang, Yaru Niu, Enna Sachdeva, Chiho Choi, Teruhisa Misu, Masayoshi Tomizuka, and Wei Zhan. Domain knowledge driven pseudo labels for interpretable goal-conditioned interactive trajectory prediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, pages 13034–13041. IEEE, 2022.

[156] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2443–2451. IEEE, 2020.

[157] Qiao Sun, Xin Huang, Junru Gu, Brian C. Williams, and Hang Zhao. M2I: from factored marginal trajectory prediction to interactive prediction. In *IEEE/CVF Con-*

ference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 6533–6542. IEEE, 2022.

[158] Ali Taalimi. Robust multi-object tracking using confident detections and safe tracklets. 09 2015.

[159] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15398–15408, 2019.

[160] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020.

[161] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[162] Li-Wu Tsao, Yan-Kai Wang, Hao-Siang Lin, Hong-Han Shuai, Lai-Kuan Wong, and Wen-Huang Cheng. Social-ssl: Self-supervised cross-sequence representation learning based on transformers for multi-agent trajectory prediction. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXII*, volume 13682 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2022.

[163] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi-Pang Lam, Dragomir Anguelov, and Benjamin Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. *CoRR*, abs/2111.14973, 2021.

[164] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[165] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *CVPR*, 2020.

[166] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 577–584. Morgan Kaufmann, 2001.

[167] Jianqiang Wang, Huang Heye, Keqiang Li, and Jun Li. Towards the unified principles for level 5 autonomous vehicles. *Engineering*, 2021.

[168] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10296–10305. Computer Vision Foundation / IEEE, 2019.

[169] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[170] Yue Wang, Alireza Fathi, Abhijit Kundu, David A. Ross, Caroline Pantofaru, Thomas A. Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, volume 12367 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2020.

[171] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *ECCV*, 2018.

[172] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75:247–266, 01 2007.

[173] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019.

[174] Xin Wu, Yi Cai, Qing Li, Jingyun Xu, and Ho-fung Leung. Combining contextual information by self-attention mechanism in convolutional neural networks for text classification. In *WISE*. Springer, 2018.

[175] Yuxin Wu and Kaiming He. Group normalization. *Int. J. Comput. Vis.*, 128(3):742–755, 2020.

[176] Zhenghao Xi, Heping Liu, Huaping Liu, and Bin Yang. Multiple object tracking using the shortest path faster association algorithm. *TheScientificWorldJournal*, 2014:481719, 08 2014.

[177] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713, 2015.

[178] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.

[179] Jiangjian Xiao, Hui Cheng, Harpreet Sawhney, and Feng Han. Vehicle detection and tracking in wide field-of-view aerial video. 2010.

[180] Guotao Xie, Hongbo Gao, Lijun Qian, Bin Huang, Keqiang Li, and Jianqiang Wang. Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models. *IEEE Trans. Ind. Electron.*, 65(7):5999–6008, 2018.

[181] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. MLCVNet: Multi-level context votenet for 3d object detection. In *CVPR*, 2020.

[182] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4606–4615. IEEE Computer Society, 2018.

[183] Kota Yamaguchi, Alexander C. Berg, Luis E. Ortiz, and Tamara L. Berg. Who are you with and where are you going? In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 1345–1352. IEEE Computer Society, 2011.

[184] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[185] Bin Yang, Ming Liang, and Raquel Urtasun. HDNet: Exploiting HD maps for 3d object detection. In *Conference on Robot Learning*, 2018.

[186] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3d object detection from point clouds. In *CVPR*, 2018.

[187] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *CVPR*, 2019.

[188] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3DSSD: Point-based 3d single stage object detector. In *CVPR*, 2020.

[189] Maosheng Ye, Tongyi Cao, and Qifeng Chen. TPCN: temporal point cloud networks for motion forecasting. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11318–11327. Computer Vision Foundation / IEEE, 2021.

[190] Maosheng Ye, Jiamiao Xu, Xunnong Xu, Tongyi Cao, and Qifeng Chen. DCMS: motion forecasting with dual consistency and multi-pseudo-target supervision. *CoRR*, abs/2204.05859, 2022.

[191] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. LiDAR-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *CVPR*, 2020.

[192] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. LiDAR-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *CVPR*, 2020.

[193] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10871–10880. PMLR, 2020.

[194] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. arxiv:1511.07122, 2015.

[195] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Deep reinforcement learning with relational inductive biases. In *ICLR*, 2018.

[196] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

[197] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 532–539. IEEE, 2021.

[198] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8660–8669. Computer Vision Foundation / IEEE, 2019.

[199] Bowen Zhang and Harold Soh. Large language models as zero-shot human models for human-robot interaction. *CoRR*, abs/2303.03548, 2023.

[200] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*. PMLR, 2019.

[201] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. 06 2008.

[202] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, volume 9907 of *Lecture Notes in Computer Science*, pages 649–666. Springer, 2016.

[203] Wenxiao Zhang and Chunxia Xiao. PCAN: 3d attention map learning using contextual information for point cloud based retrieval. *CoRR*, abs/1904.09793, 2019.

[204] Zhizheng Zhang, Cuiling Lan, Wenjun Zeng, Xin Jin, and Zhibo Chen. Relation-aware global attention for person re-identification. In *CVPR*, 2020.

[205] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. TNT: target-driven trajectory prediction. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 895–904. PMLR, 2020.

[206] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019.

[207] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

[208] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection, 2019.

[209] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. SSN: shape signature networks for multi-class object detection from point clouds. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, volume 12370 of *Lecture Notes in Computer Science*, pages 581–597. Springer, 2020.

[210] ore Petrović, Radomir Mijailović, and Dalibor Peić. Traffic accidents with autonomous vehicles: Type of collisions, manoeuvres and errors of conventional vehicles' drivers. *Transportation research procedia*, 45:161–168, 2020.

Part IV

APPENDICES

Appendix A

SA-Det3D: Implementation Details and Supplementary Results

In this document, we provide technical details and additional experimental results for SA-Det3D.

A.1 Network Architectures and Training Details

A.1.1 Architectural details

The detailed specification of the various layers in our FSA and DSA augmented baselines—PointPillars [83], SECOND [184], Point-RCNN [144] and PV-RCNN [143]—is documented in Tab. A.2, Tab. A.3, Tab. A.4, and Tab. A.5, respectively. We also provide the details of a reduced parameter baseline that aims to compare the performance of the model with similar number of parameters and FLOPs compared to their FSA and DSA counterparts.

A.1.2 Experimental settings

Additional details on encoding, training, and inference parameters are as follows. For pillar and voxel-based detection, we use absolute-position encoding for the full self-attention blocks [164]. For the *test* submissions to KITTI and nuScenes official servers, we retain the full parameterization of the original baselines. For the nuScenes *test* submission, we follow the configuration for PP described in Tab. A.2, while adding the DSA module with

Backbone	Batch Size	Start LR	Max LR
PointPillars	16	0.0003	0.003
SECOND			
Point-RCNN	8	0.001	0.01
PV-RCNN			

Table A.1: Batch size and learning rate configurations for each backbone model on KITTI benchmark

4 heads, 2 layers, 64-dimensional context, 2 m deformation radius, 4096 sampled pillars and an up-sampling method described in the following subsection. For the Waymo Open dataset *validation* evaluation, we use the configuration for DSA-SECOND as described in Tab. A.3, except that we use 4096 sampled keypoints, 2 m deformation radius and 1 m interpolation radius. We use Pytorch [121] and the recently released OpenPCDet [160] repository for our experiments. Our models are trained from scratch in an end-to-end manner with the ADAM optimizer [75]. The learning rates used for the different models are given in Tab. A.1. For the proposal refinement stage in two-stage networks [144], [143], we randomly sample 128 proposals with 1:1 ratio for positive and negative proposals. A proposal is considered positive if it has at-least 0.55 3D IoU with the ground-truth boxes, otherwise it is considered to be negative. For inference, we keep the top-500 proposals generated from single stage approaches [83], [184] and the top-100 proposals generated from two stage approaches [144], [143] with a 3D IoU threshold of 0.7 for non-maximum-suppression (NMS). An NMS classification threshold of 0.1 is used to remove weak detections.

Attribute	PP [83]	PP$_{red}$	FSA-PP	DSA-PP
Layer: 2D CNN Backbone				
Layer-nums	[3, 5, 5]	[3, 5, 5]	[3, 5, 5]	[3, 5, 5]
Layer-stride	[2, 2, 2]	[2, 2, 2]	[2, 2, 2]	[2, 2, 2]
Num-filters	[64, 128, 256]	[64, 64, 128]	[64, 64, 64]	[64, 64, 64]
Upsample-stride	[1, 2, 4]	[1, 2, 4]	[1, 2, 4]	[1, 2, 4]
Num-upsample-filters	[128, 128, 128]	[128, 128, 128]	[128, 128, 128]	[128, 128, 128]
Layer: Self-Attention				
Stage Added	-	-	Pillar feature	Pillar feature
Num layers	-	-	2	2
Num heads	-	-	4	4
Context Linear Dim	-	-	64	64
Num Keypoints	-	-	-	2048
Deform radius	-	-	-	3.0m
Feature pool radius	-	-	-	2.0m
Interpolation MLP Dim	-	-	-	64
Interpolation radius	-	-	-	1.6m
Interpolation samples	-	-	-	16

Table A.2: Architectural details of PointPillars [83], our reduced parameter PointPillars version, proposed FSA-PointPillars and DSA-PointPillars

Attribute	SECOND [184]	SECOND$_{red}$	FSA-SECOND	DSA-SECOND
Layer: 3D CNN Backbone				
Layer-nums in Sparse Blocks	[1, 3, 3, 3]	[1, 3, 3, 2]	[1, 3, 3, 2]	[1, 3, 3, 2]
Sparse tensor size	128	64	64	64
Layer: 2D CNN Backbone				
Layer-nums	[5, 5]	[5, 5]	[5, 5]	[5, 5]
Layer-stride	[1, 2]	[1, 2]	[1, 2]	[1, 2]
Num-filters	[128, 256]	[128, 160]	[128, 128]	[128, 128]
Upsample-stride	[1, 2]	[1, 2]	[1, 2]	[1, 2]
Num-upsample-filters	[256, 256]	[256, 256]	[256, 256]	[256, 256]
Layer: Self-Attention				
Stage Added	-	-	Sparse Tensor	Sparse Tensor
Num layers	-	-	2	2
Num heads	-	-	4	4
Context Linear Dim	-	-	64	64
Num Keypoints	-	-	-	2048
Deform radius	-	-	-	4.0m
Feature pool radius	-	-	-	4.0m
Interpolation MLP Dim	-	-	-	64
Interpolation radius	-	-	-	1.6m
Interpolation samples	-	-	-	16

Table A.3: Architectural details of SECOND [184], our reduced parameter SECOND version, and proposed FSA-SECOND and DSA-SECOND

Attribute	Point-RCNN [144]	Point-RCNN$_{red}$	FSA-Point-RCNN	DSA-Point-RCNN
Layer: Multi-Scale Aggregation				
N-Points	[4096, 1024, 256, 64]	[4096, 1024, 256, 64]	[4096, 1024, 256, 64]	[4096, 1024, 128, 64]
Radius	[0.1, 0.5], [0.5, 1.0], [1.0, 2.0], [2.0, 4.0]	[0.1, 0.5], [0.5, 1.0], [1.0, 2.0], [2.0, 4.0]	[0.1, 0.5], [0.5, 1.0], [1.0, 2.0], [2.0, 4.0]	[0.1, 0.5], [0.5, 1.0], [1.0, 2.0], [2.0, 4.0]
N-samples	[16, 32]	[16, 32]	[16, 32]	[16, 32]
MLPs	[16, 16, 32], [32, 32, 64], [64, 64, 128], [64, 96, 128] [128, 196, 256], [128, 196, 256] [256, 256, 512], [256, 384, 512]	[16, 32], [32, 64], [64, 128], [64, 128] [128, 256], [128, 256] [256, 512], [256, 512]	[16, 32], [32, 64], [64, 128], [64, 128] [128, 256], [128, 256] [256, 512], [256, 512]	[16, 32], [32, 64], [64, 128], [64, 128] [128, 256], [128, 256] [256, 512], [256, 512]
FP-MLPs	[128, 128], [256, 256], [512, 512], [512, 512]	[128, 128], [128, 128], [128, 128], [128, 512]	[128, 128], [128, 128], [128, 128], [128, 128]	[128, 128], [128, 128], [128, 128], [128, 128]
Layer: Self-Attention				
Stage Added	-	-	MSG-3 and MSG-4	MSG-3 and MSG-4
Num layers	-	-	2	2
Num heads	-	-	4	4
Context Linear Dim	-	-	64	64
Num Keypoints	-	-	-	(128, 64)
Deform radius	-	-	-	(2.0, 4.0)m
Feature pool radius	-	-	-	(1.0, 2.0)m
Interpolation MLP Dim	-	-	-	(64, 64)
Interpolation radius	-	-	-	(1.0, 2.0)m
Interpolation samples	-	-	-	(16, 16)

Table A.4: Architectural details of Point-RCNN [144], our reduced parameter Point-RCNN version, proposed FSA-Point-RCNN and DSA-Point-RCNN

Attribute	PV-RCNN [143]	FSA-PVRCNN	DSA-PVRCNN
Layer: 3D CNN Backbone			
Layer-nums in Sparse Blocks	[1, 3, 3, 3]	[1, 3, 3, 2]	[1, 3, 3, 3]
Sparse tensor size	128	64	128
Layer: 2D CNN Backbone			
Layer-nums	[5, 5]	[5, 5]	[5, 5]
Layer-stride	[1, 2]	[1, 2]	[1, 2]
Num-filters	[128, 256]	[128, 128]	[128, 256]
Upsample-stride	[1, 2]	[1, 2]	[1, 2]
Num-upsample-filters	[256, 256]	[256, 256]	[256, 256]
Layer: Self-Attention			
Stage Added	-	Sparse Tensor and VSA	VSA
Num layers	-	2	2
Num heads	-	4	4
Context Linear Dim	-	128	128
Num Keypoints	-	-	2048
Deform radius	-	-	[0.4, 0.8], [0.8, 1.2], [1.2, 2.4], [2.4, 4.8]
Feature pool radius	-	-	Multi-scale: (0.8, 1.6)m
Interpolation MLP Dim	-	-	Multi-scale: (64, 64)
Interpolation radius	-	-	Multi-scale: (0.8, 1.6)m
Interpolation samples	-	-	Multi-scale: (16, 16)

Table A.5: Architectural details of PV-RCNN [143], and proposed FSA-PVRCNN and DSA-PVRCNN

A.2 Detailed Results

We provide additional experimental details on the validation split of the KITTI [43] dataset in this section. In Tab. A.6, we first show the 3D and BEV AP for moderate difficulty on the *Cyclist* class for PV-RCNN and its variants. The table shows that both our proposed modules improve on the baseline results. This showcases the robustness of our approach in also naturally benefiting smaller and more complicated objects like cyclists. We then proceed to list the 3D AP and BEV performances with respect to distance from the ego-vehicle in Tab. A.7. We find that the proposed blocks especially improve upon detection at further distances, where points become sparse and context becomes increasingly important. These results hold especially for the *cyclist* class - as opposed to the *car* class, which shows that context is possibly more important for smaller objects with reduced number of points available for detection. In Tab. A.8, we provide results for all three difficulty categories for the *car* class. We see consistent improvements across backbones with various input modalities on the *hard* category. This is consistent with our premise that samples in the hard category can benefit more context information of surrounding instances. We also note that PointPillars [83], which loses a lot of information due to pillar-based discretization of points, can supplement this loss with fine-grained context information.

	3D	BEV
PV-RCNN [143]	70.38	74.5
PV-RCNN + DSA	**73.03**	**75.45**
PV-RCNN + FSA	71.46	74.73

Table A.6: Performance comparison for moderate difficulty cyclist class on KITTI *val* split.

Distance	Model	Car	Cyclist	Pedestrian
0-30m	PV-RCNN [143]	91.71	73.76	56.82
	DSA	91.65	**74.89**	59.61
	FSA	**93.44**	74.10	**61.65**
30-50m	PV-RCNN [143]	50.00	35.15	-
	DSA	52.02	**47.00**	-
	FSA	**52.76**	39.74	-

Table A.7: Comparison of nearby and distant-object detection on the moderate level of KITTI *val* split with AP calculated by 40 recall positions

Model	Modality	Params (M)	GFLOPs	Car 3D AP Easy	Car 3D AP Moderate	Car 3D AP Hard
PP [83]	BEV	4.8	63.4	87.75	78.39	75.18
PP$_{red}$	BEV	1.5	**31.5**	88.09	78.07	75.14
PP-DSA	BEV	1.1	32.4	89.37	78.94	75.99
PP-FSA	BEV	**1.0**	31.7	**90.10**	**79.04**	**76.02**
SECOND [184]	Voxel	4.6	76.7	90.55	81.61	78.61
SECOND$_{red}$	Voxel	2.5	**51.2**	89.93	81.11	78.30
SECOND-DSA	Voxel	2.2	52.6	**90.70**	**82.03**	**79.07**
SECOND-FSA	Voxel	**2.2**	51.9	89.05	81.86	78.84
Point-RCNN [144]	Points	4.0	27.4	**91.94**	80.52	78.31
Point-RCNN$_{red}$	Points	2.2	24.1	91.47	80.40	78.07
Point-RCNN-DSA	Points	**2.3**	**19.3**	91.55	81.80	79.74
Point-RCNN-FSA	Points	2.5	19.8	91.63	**82.10**	**80.05**

Table A.8: Detailed comparison of 3D AP with baseline on KITTI *val* split with 40 recall positions

Appendix B

SSL-Lanes: Implementation Details and Supplementary Results

In this document, we provide technical details and additional experimental results for SSL-Lanes.

B.1 Detailed Network Architecture for Baseline

We provide the detailed network architecture of our baseline in Fig. B.1.
For the *agent feature extractor*, the architecture is similar to [88]. We use an 1D CNN to process the trajectory input. The output is a temporal feature map, whose element at $t = 0$ is used as the agent feature. The network has three groups/scales of 1D convolutions. Each group consists of two residual blocks [56], with the stride of the first block as 2. Feature Pyramid Network (FPN) [95] fuses the multi-scale features, and applies another residual block to obtain the output tensor. For all layers, the convolution kernel size is 3 and the number of output channels is 128. Layer normalization [6] and Rectified Linear Unit (ReLU) are used after each convolution.
The *map feature extractor* has two LaneConv residual [56] blocks which are the stack of a LaneConv(1, 2, 4, 8, 16, 32) and a linear layer, as well as a shortcut. All layers have 128 feature channels. Layer normalization [6] and ReLU are used after each LaneConv and linear layer.
For the map-aware agent feature (M2A) module, the distance threshold is 12m. It is 100m for the agent-to-agent (A2A) interaction module. The two *interaction modules* have two

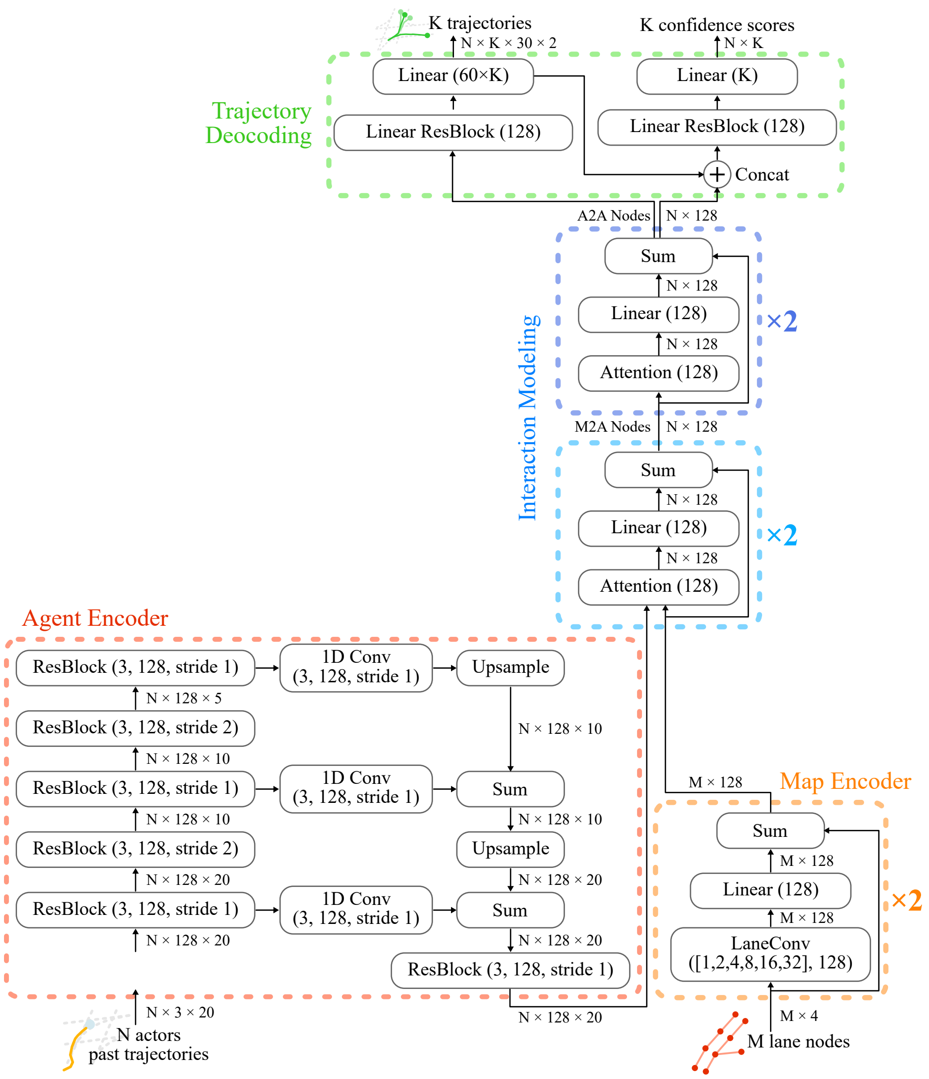


Figure B.1: Architecture of the baseline model

residual blocks, which consist of a stack of an attention layer and a linear layer, as well as a residual connection. All layers have 128 output feature channels.
Taking the interaction-aware actor features as input, our *trajectory decoder* is a multi-modal prediction header that outputs the final motion forecasting. For each agent, it predicts K possible future trajectories and confidence scores. The header has two branches, a regression branch to predict the trajectory of each mode and a classification branch to predict the confidence score of each mode.
Key differences with Lane-GCN [88]: Our main difference is we use two Lane-Conv blocks instead of four as map-feature extractor in order to prevent over-smoothing in GNNs [133]. We also do not use the four-way fusion proposed by Lane-GCN and do away with the agent to map (A2M) and the map to map (M2M) interaction blocks, which saves compute and memory.

B.2 Implementation of Pretext Tasks

In this section, we discuss various design decisions for the proposed pretext tasks.

B.2.1 Lane-Masking

For this pretext task, we mask m_a percent of every lane and reconstruct its features. In

Method	**m$_a$**	**minADE$_6$**	**minFDE$_6$**	**MR$_6$**
Baseline	-	0.73	1.12	11.07
Random Masking	0.4	0.71	1.03	9.11
Lane-Masking	0.3	0.71	1.04	9.02
Lane-Masking	0.4	**0.70**	**1.02**	**8.84**
Lane-Masking	0.5	0.71	1.05	9.31

Table B.1: Effect of masking ratio (m_a) on forecasting performance for lane-masking task

Tab. B.1, we study the influence of masking ratio on the final forecasting performance. Random masking refers to masking out m_a percent random map nodes and lane-masking refers to masking out m_a percent of lanes in the map. We finally choose $m_a = 0.4$ as the most effective parameter for the lane-masking pretext task, which outperforms random masking. The model infers missing lane-nodes to produce plausible outputs during reconstruction. We hypothesize that this reasoning is linked to learning useful representations.

B.2.2 Distance to Intersection

For this pretext task, we explore two different options for framing the problem of predicting the distance to the nearest intersection node in Tab. B.2. We first explore predicting this distance as a classification task. We group the lengths into four categories: $d_{ij} = 1$, $d_{ij} = 2$, $d_{ij} = 3$, $d_{ij} = 4$ and $d_{ij} >= 5$. We however find that this is harder to optimize than the regression loss proposed in Eq. (5.7), which we finally choose as our loss for the distance to intersection pretext task.

Method	**Pretext Loss**	**minADE$_6$**	**minFDE$_6$**	**MR$_6$**
Baseline	-	0.73	1.12	11.07
Distance to Intersection	Classification	0.72	1.06	9.64
Distance to Intersection	Regression	**<u>0.71</u>**	**<u>1.04</u>**	**<u>8.93</u>**

Table B.2: Effect of pretext loss type on forecasting performance for distance to intersection task

B.2.3 Maneuver Classification

For this pretext task, we first divide the lateral and longitudinal maneuvers by choosing a threshold angle of 20° from the vertical. We next find that constrained k-means [166] on agent end-points for lateral and longitudinal maneuvers works best to separate the trajectory samples into different clusters. This is illustrated in Fig. B.2. For differentiating the longitudinal maneuvers from the lane-change maneuver, we check a combination of the distance from the lane centerlines for start and stop positions and the orientations of the nearest centerline for start and stop positions.

B.2.4 Success/Failure Classification

For this pretext task, the primary bottleneck is the fact that the number of positive examples if far fewer than the number of negative examples. This is because there are only a few success examples in a 2m area near the end-point of a single recorded ground-truth trajectory, while the rest of the points in the scene can be considered as failure examples. We consider first setting $\epsilon = 3$m, i.e. a wider area for success examples, and then reducing it to $\epsilon = 2$m linearly over the total number of training steps. We find that this can actually

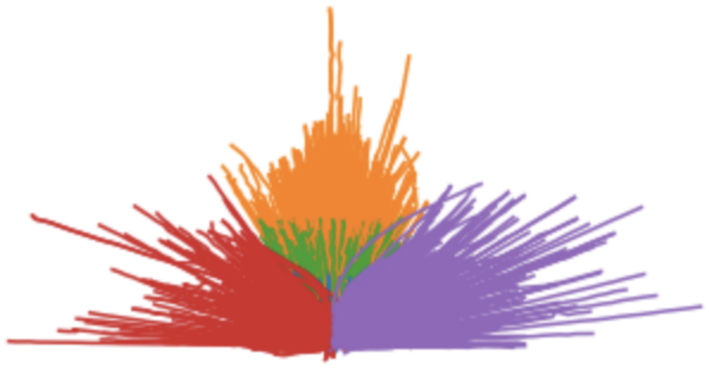


Figure B.2: Modes of driving from unsupervised clustering of data

harm the final forecasting performance. We thus follow [206] to use focal loss to train our auxiliary classification task.

B.3 Qualitative Results

We first present some multi-modal prediction trajectories on several hard cases shown in Fig. 5.4. SSL-Lanes can capture left and right turns better, while also being able to discern acceleration at intersections. Our pretext tasks provide priors for the model and provides data-driven regularization for free. This can improve forecasting because of better understanding of map topology, agent context with respect to the map, and also improve generalization for maneuver imbalance implicitly present in data. We next provide more visual results of our proposed SSL-Lanes on the Argoverse validation set in Fig. B.3. Generally, these qualitative results demonstrate the effectiveness of our proposed pretext tasks.

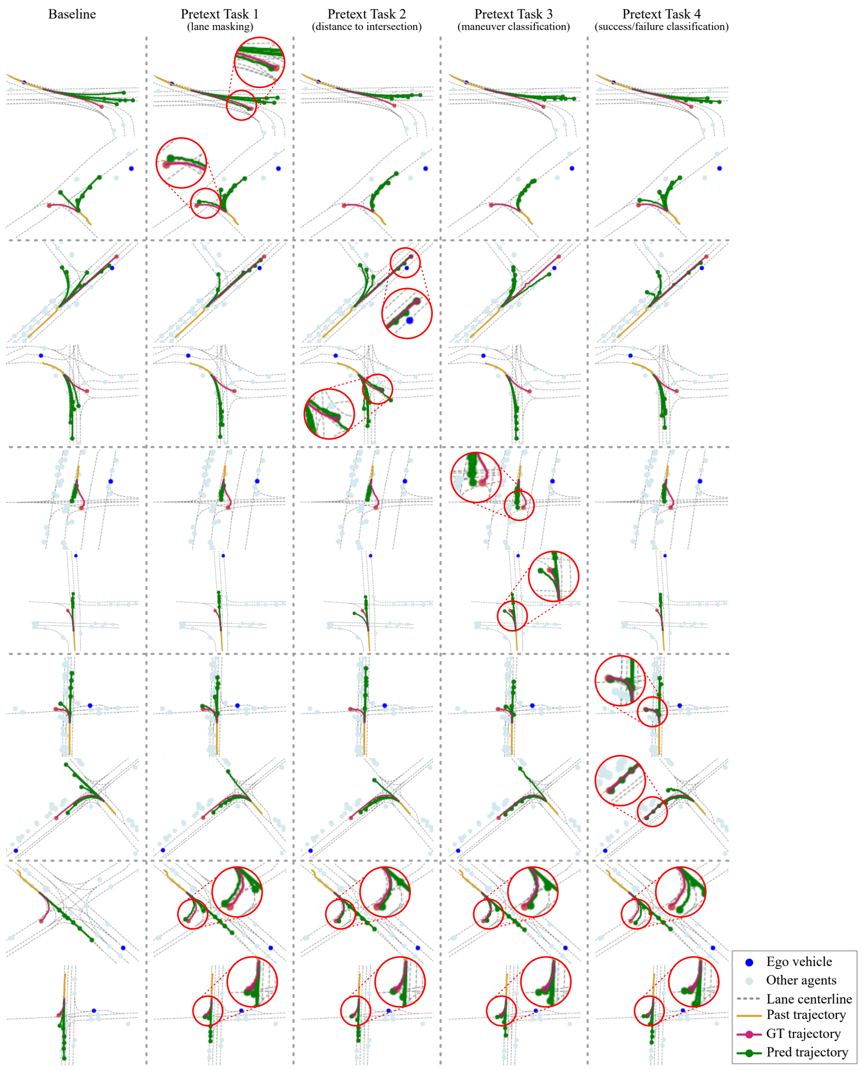


Figure B.3: Qualitative results for our proposed SSL-Lanes pretext tasks on the Argoverse [26] validation set. We outperform the baseline on several difficult cases at intersections and lane-changes.

B.4 Discussion: Choice of Dataset

We now compare the commonly used motion-forecasting datasets, i.e., nuScenes [19], Waymo-Open-Motion-Dataset (WOMD) [38] and Argoverse [26]. We individually discuss why Argoverse is best positioned to bring out the benefits of our proposed work.

- *Scale of Data:* We first compare the dataset size and diversity. We note that Argoverse is not only larger and more diverse than nuScenes, but also has greater number of training samples and unique trajectories compared to WOMD.

	nuScenes	WOMD	Argoverse
Number of Unique Tracks:	4.3k	7.65m	11.7m
Number of Training Segments:	1k	104k	324k

- *Interesting Scenarios for Forecasting Evaluation:* We next compare if the datasets specifically mines for interesting scenarios, which is the area we want to improve the current baseline. nuScenes was not collected to capture a wide diversity of complex and interesting driving scenarios. WOMD on the other hand specifically mines for pairwise interaction scenarios, where the main objective is to improve forecasting for interacting agents. However, the scope of our study is to primarily focus on motion at intersections undergoing lane-changes and turns. We expect the SSL-losses to improve understanding of the context/environment, trajectory embeddings and address data-imbalance w.r.t. maneuvers. We leave heavy interaction-based use cases for future work. Finally, Argoverse mines for interesting motion patterns at intersections, which involve lane-changes, acceleration/deceleration, and turns. We thus find this dataset best suited to showcase our proposed method.

- *Community focus on Argoverse:* We also find that many popular motion forecasting methods published by the robotics community have also included evaluations only on the Argoverse dataset including: Lane-GCN, Lane-RCNN, PRIME, DCMS, TPCN, mm-Transformer, HiVT, Multi-modal Transformer, DSP etc. This makes it easier for us to position our work with respect to these approaches.

B.5 Discussion: Potential of this Work

We expect this work to influence real world deployment of SSL forecasting methods for autonomous driving. Another use case for this work is realistic behavior generation in traffic simulation. The general construction of the prediction problem, inspired by [88], enables a generic understanding of how an object moves in a given environment without memorizing the training data. A neural network may learn to associate particular areas of a scene with certain motion patterns. To prevent this, we centre around the agent of interest and normalize all other trajectory and map coordinates with respect to it. We predict relative motion as opposed to absolute motion for the future trajectory. This helps to learn general motion patterns. Reconstructing the map or predicting distances from map elements are conducted in a frame-of-reference relative to the agent of interest. This helps in learning general map connectivity. Following work in pedestrian trajectory prediction, we also additionally add random rotations to the training trajectories to reduce directional bias. Furthermore, we provide strong evidence that SSL-based tasks provide better generalization compared to pure supervised training, thereby having the ability to effectively reuse the same prediction model across different scenarios.