Deep Learning Solutions for Perception and Motion Forecasting in Autonomous Vehicles

Burak Alp Inan

Supervisor: Prof. Nabil Aouf

Department of Engineering
City, University of London

This dissertation is submitted for the degree of
Doctor of Philosophy

November 2024

Acknowledgements

I would like to begin by sincerely thanking my supervisor, Prof.Nabil Aouf, for his outstanding mentorship, support, and guidance throughout the course of this research. His dedication and insight have been invaluable, and I am deeply grateful for his contributions to my academic growth.

I would also like to thank Dr. Duarte Rondao for his support and for the insightful discussions and ideas we shared throughout the course of this research. His input has been valuable in shaping various aspects of this work, and I greatly appreciate his collaboration.

I would like to acknowledge my fellow researchers and colleagues in the RAMI group. The discussions we had and the shared environment contributed to my progress, and I appreciate their support throughout this journey.

I would also like to extend my thanks to my close friend Ramazan Enisoglu for always being there for me, providing encouragement and moral support when I needed it the most.

I gratefully acknowledge the funding received towards my PhD from the Ministry of Education of Turkiye. Without their support this research would not be possible.

I would also like to extend my gratitude to my father-in-law, Dr. Abdullah, and my mother-in-law, Nursel for always being there for me and and their consistent support and encouragement throughout my PhD journey.

To my father, Prof. Kenan, my mother, Aysegul, and my sister, Munevver, I am profoundly thankful for your unwavering support, guidance, and understanding. Your belief in me through all the highs and lows has been invaluable, and I am forever grateful for your presence and love in my life.

Finally, and most importantly, I would like to extend my deepest gratitude to my beloved wife, Aysegul, and our baby son, Yigit Eymen. Your endless love, patience, and support have been my greatest source of strength throughout this journey. Having you both by my side, especially during this transformative time, has been a blessing, and I could not have achieved this without you.

Abstract

The major challenges for autonomous driving systems are the need for accurate and real-time perception, tracking, and motion forecasting necessary to navigate safely through complex and dynamic environments. These systems must be able to intuitively make decisions within split seconds in an urban environment with multiple agents interacting. This thesis proposes a comprehensive framework that addresses the LiDAR-based segmentation challenges, multi-object tracking, and trajectory forecasting with a focus on enhancing its accuracy, robustness, and efficiency.

The contribution of this work is to develop a hybrid approach to the segmentation of LiDAR, which would integrate synthetic data with real-world data sets. Synthetic data, created through simulated environments, may allow the model to experience different scenarios that could not be fully captured with real-world data alone. This combination enhances the generalization capability of the segmentation models, which can then handle difficult situations, occlusions, and variations in object density and wide light condition variations. Apart from that, the incorporation of multi-scale feature extraction methods helps in processing fine-grained details over various spatial resolution levels. This significantly improves accuracy in segmentation without choking compromising efficiency. The hybrid approach ensures that the model performance of segmentation is good enough to work on various different urban driving environments.

Besides these improvements, this thesis further exploits the efficacy of Vision Transformers for segmenting LiDAR point clouds. An attention mechanism introduced within the ViT captures both local and global geometric features much better than traditional convolutional networks. Employing transformers make the segmentation model understand much better the relationships between points. This would also successfully improve accuracy in object detection and classification, especially in complex sets of data. This significantly enhances the probability that the segmentation model will find and classify various objectsin real-time . In this respect, it is very effective in highly demanding situations where the need for an accurate understanding of the environment is required, such as at busy street intersections or on highways.

The thesis also addresses the need to have good object tracking in dynamic surroundings by proposing a transformer-based multi-object tracking framework. It leverages a joint 2D-3D sensor fusion framework that fuses LiDAR and camera data so as to further enhance the accuracy in tracking of dynamic agents. Together, exploiting these sensing modalities allows the tracker to appreciate depth and geometric information given by LiDAR and the rich visual details provided by camera images. The system is designed to facilitate accurate tracking of multiple agents in real-time, which is crucial for ensuring safety and effective decision-making in autonomous driving systems.

Building on the improvements of segmentation and tracking, this thesis proposes IRMTR, a conveniently novel approach to multi-agent motion forecasting. IRMTR framework employs anchored goal queries and Gaussian Mixture Model that serve to generate intention points representing the most likely positions an agent is willing to take in the near future. These points of intentions are then fine-tuned using a hybrid local-global query mechanism to improve the predictive outcome of the model for future trajectories generated by dynamic agents in real-

time. Considering that the IRMTR model updates its predictions by incorporating both the local interaction among neighboring agents and the global context of the driving environment, it boosts trajectory prediction accuracy quite significantly, especially in complicated traffic conditions. In fact, this model turns out to be particularly effective in forecasts of behaviors that involve lane changes, merging, and interaction at intersections where accurate trajectory forecast should be effective to enable collision avoidance and proactive navigation.

Overall, this thesis contributes to the advancement of autonomous vehicle technologies by proposing novel methods for LiDAR segmentation, multi-object tracking, and motion forecasting. By combining state-of-the-art deep learning techniques, this research lays the groundwork for future developments in real-time mapping and decision-making systems in autonomous vehicles.

Keywords— Autonomous driving, Scene understanding, Synthetic data, LiDAR segmentation, Transformers, Vision transformers, Object tracking, Motion forecasting

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence.

ANN Artificial Neural Network.

AP Average Precision.

BPTT Backpropagation Through Time.

CNN Convolutional Neural Network.

CRF Conditional Random Field.

DETR Detection Transformer.

DNN Deep Neural Network.

FC Fully Connected.

FFN Feed Forward Network.

GMM Gaussian Mixture Model.

GPS Global Positioning System.

GPU Graphics Processing Unit.

HD High Definition.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

INS Inertial Navigation System.

IoU Intersection-over-Union.

IRMTR Intention Refined Motion Transformer for Motion Forecasting.

LiDAR Light Detection and Ranging.

LSTM Long-short Term Memory.

mIoU mean Intersection-over-Union.

ML Machine Learning.

MLP Multi-layer Perceptron.

MOT Multiple Object Tracking.

MOTA Multi-Object Tracking Accuracy.

NLP Natural Language Processing.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

SAC Spatially-Adaptive Convolution.

Seq2Seq Sequence to Sequence.

SGD Stochastic Gradient Descent.

ViT Vision Transformers.

CHAPTER 1

Introduction

In this opening chapter, the reader is introduced to the current context that gives rise to this work and is provided with a summary of how deep learning methods, whether adopted or devised within this study, are integrated with autonomous driving applications to tackle challenges in the field.

1.1 Motivation

Fully autonomous vehicle technology is rapidly developing and promises a future of transportation that is safer, less congested, and more mobile across wider sections of the population. At the very heart of the drive to create fully autonomous vehicles—nowhere near to realization due to the subtlety of natural driving scenarios—lies an accurate perception of the surroundings and the prediction of the future motion of surrounding objects.

To this end, perception and motion forecasting are two of the critical pieces in the overall decision-making for an autonomous vehicle. Perception concerns the processing of raw data streams from sensors like cameras, Light Detection and

Ranging (LiDAR), and radar for object detection and classification tasks, related to pedestrians and cyclists, other cars, etc. The process of motion forecasting extends into predicting the future trajectories of such dynamic agents, thereby allowing the vehicle to act in anticipation against possible dangers.

Conventional methods of perception and motion forecasting involve hand-crafted features coupled with rule-based systems. These do not generalize easily to the large variations and unpredictability found on the road. In such complicated scenarios, where variability is very high, for example, in an urban area with very dense flow of traffic, highly changeable weather, and occlusions, these algorithms can perform poorly. Limitations of traditional methods show that there is a strong desire for more robust and adaptive solutions that will handle the much finer details associated with dynamic driving environments.

Among these are deep learning, a subset of artificial intelligence, which uses neural networks normally built with multiple layers. Deep learning models can learn hierarchical representations of data. Such models become effective in object detection and easily classify them, with good trajectory prediction in relation to sensor inputs. Consider Convolutional Neural Networks (CNNs) in recognizing images; they have transformed these tasks into seamless and ideal tasks for processing data obtained from cameras. Similar characteristics have also allowed Recurrent Neural Networks (RNNs) and Long-short Term Memorys (LSTMs) to handle sequential data with much ease and hence seem to be a fitting choice for motion forecasting tasks.

Deep learning, integrated into perception systems, allows the autonomous vehicle to better grasp richer and more complex scenes. For example, semantic segmentation networks perform classification on a per-pixel basis in an image. This gives a very high level of detail in environment understanding. Deep learning models in motion forecasting have proven to learn from large trajectory data and

predict the future movement of dynamic agents with high accuracy. These functionalities are some of the ones necessary to maintain safety and efficiency while navigating through unpredictable environments.

However, deep learning models themselves face challenges when being used in real-world self-driving car applications. First, real-time processing is indispensable, and the models have to be efficient enough to execute on an embedded system with limited computational resources. Another challenge is the robustness of models against changing sensor data due to changes in lighting conditions, weather, and sensor noise. Ensuring safety and reliability, these models will need rigorous testing and validation for operation in all sorts of driving scenarios.

Apart from this, deep learning models require a vast breach in the dataset for training. Indeed, it is costly and time-consuming to acquire and annotate enough real-world data that covers the huge variation in driving conditions. Therefore, synthetic data from state-of-the-art driving simulators could complement real-world data through offering several variations in scenarios against which training and testing may take place. Synthetic data allows one to vary environmental conditions and traffic situations in a controlled manner, hence giving an increase in the generalization capability of the model to unseen situations.

This thesis will develop deep learning solutions to such challenges in accuracy, efficiency, and robustness in perception and motion forecasting within the autonomous vehicle arena. We aim to improve environmental perception capability and the vehicle's action prediction of other agents through designing advanced neural network architectures that could be executed in real time. Our approach leverages better utilisation of multi-sensor data fusion by using cameras, LiDAR together to enhance richness in input data itself, hence improving model performance. We also develop methods to improve the generalization capabilities of our models by incorporating synthetic data into more varied training datasets. Valida-

tion of our models on both real-world and simulated data is a meaningful attempt at ensuring that our models work on various environments and conditions.

With this work, we would like to make another step forward in autonomous vehicle technology and provide an effective deep learning-based solution for perception and motion forecasting. Our work bridges the gap from theoretical research to practical application, with the goal of establishing an accurate, efficient, and reliable system in realistic conditions. The motivation above forms the cornerstone for exploring innovative deep learning approaches through which the autonomous vehicle can safely and intelligently move, thus contributing to the big picture of intelligent transportation systems.

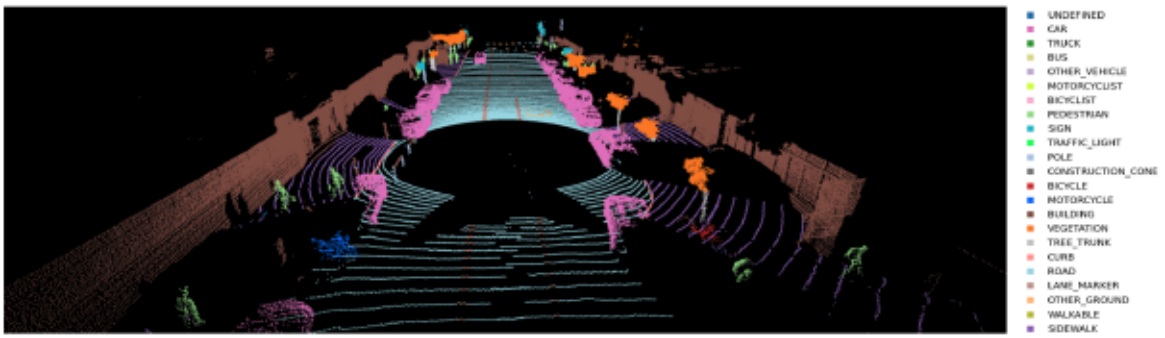

Figure 1.1: 3D LiDAR semantic mapping of objects Behley et al. (2019).

1.2 Perception

Perception can be said to be one end of the autonomous vehicle systems; it needs interpretation of sensor data to create an understanding of the real-time environment of the vehicle. It basically encompasses object detection and classification, estimation of its position and velocity, prediction of future movements, thereby generating a comprehensive understanding of surroundings necessary for safe navigation. Hence, proper perception enables an autonomous vehicle to de-

termine its route, avoid obstacles, and deal with complex situations successfully through proper response to dynamic changes from moving vehicles, pedestrians, and other unexpected obstacles.

The major perception-related challenges with autonomous vehicles will pertain to processing speed and computational load, data integration from multiple sensors, accuracy and precision, handling of dynamic environments, scalability, and robustness in adverse conditions. This will necessarily involve the processing of huge amounts of sensor data in the shortest time possible so that the representation of the world around is kept accurate and updated. The autonomous car relies on several sensors, including LiDAR, cameras, and radar, supplying data about the environment. Integrating information from such diverse sources into one coherent perception is complex and computationally intensive. The accuracy of such a perception system has to be very high for safe navigation; minor errors may lead to big issues such as collisions or failures in navigation. The environmental setting of the vehicle keeps on changing, and perception systems are expected to detect and predict the motion of dynamic objects in real time. Moreover, it needs to be highly scalable in operation, from dense urban traffic to sparse rural roads, and must operate reliably in those adverse conditions that can significantly impair sensor performance and pose significant challenges for perception algorithms. A point cloud with annotations can be seen in Figure 1.1.

Current perception techniques in autonomous vehicles include LiDAR-based, camera-based, radar-based, and sensor fusion-based. LiDAR sensors provide high-resolution 3D point clouds that allow for very accurate object detection and classification, along with very detailed mapping of the environment. LiDAR data processing generally consists of methods such as voxel grid mapping, segmentation of point clouds, and registration of point clouds. However, most of these methods are computationally expensive and may not handle real-time processing and dynamic

objects quite as well.

Cameras deliver substantial information in the form of visual perspectives and are highly relevant for object detection and classification. Generally, CNNs are applied to image data processing and have a very high performance record in computer vision tasks. Cameras are cheap compared to LiDAR sensors but offer color information. However, their effectiveness may degrade due to variable lighting conditions while lacking in depth information.

While radar sensors provide low resolution compared to LiDAR and cameras, the advantage is that they usually work in bad weather conditions and give the information about the position and velocity. Generation of data from multiple sensors will enhance the accuracy and robustness of perception. Sensor fusion can be realized with Kalman filtering and deep learning-based fusion techniques by fusing data from LiDAR, cameras, and radar in such a way that the relative strengths of each sensor type are harnessed while mitigating respective individual limitations. These, however, have elaborate algorithms and take up much computational time. While there have been developments, the current perception methods still lack the processing speed and dynamic environment handling and much-needed accuracy for the safe envelope of navigation on the road in all conditions. This calls for fresh perspectives that should take on these challenges and others not mentioned with guaranteed perception under all scenarios. Deep learning and Artificial Intelligence (AI) techniques offer promising solutions to these challenges. We will be able to create much more efficient and capable perception systems by using neural networks and sophisticated algorithms. The AI-driven approaches allow high-speed and high-accuracy processing of huge volumes of data, data integration across diverse sensors, and dynamic changes in the environment. For instance, object detection and classification inside images and point clouds can be handled by neural networks, and even predict future motions of dynamic

objects. These are essential in helping achieve a strong perception system needed for the safety and efficiency of an autonomous vehicle.

1.3 Object Tracking

Object tracking is one of the main technologies in intelligent vehicle environment perception. Features and contextual information obtained in an initial frame help this technology in continuously positioning an object across multiple frames. In autonomous driving, the main traffic participants have to be tracked for safe navigation, such as pedestrians, cars, buses, and bicycles. Accurate and efficient object-tracking systems update the movement of surrounding objects in real time to provide the autonomous vehicle with an idea of behavioral intention that allows the latter to predict future trajectories. A simple flow diagram of object tracking is given in Figure 1.2.

Figure 1.2: Simple object tracking diagram.

Visual information represents about 90% of the environmental data on which intelligent vehicles rely for making decisions and planning. Multiple Object Tracking (MOT) is thus one of those key concepts when multiple-object trajectories are established and maintained all at once. In other words, under an optimal estimation framework, MOT algorithms correlate objects detected in an initial frame with their appearances in subsequent frames to create, in essence, a continuous tracking path for each object. Unlike single-object tracking, in MOT multiple objects are tracked simultaneously by assigning an ID to each object and maintaining

it during the sequence.

With the rapid development of sensor technologies, cameras, LiDAR, radar, and ultrasonic sensors are on-board an intelligent vehicle. This raw data is captured from heterogeneous sensors, which compensate for advantages and disadvantages to realize real-time three-dimensional environmental perception. Later, multi-sensor fusion provides a much more faithful understanding of surroundings and lays a robust basis for object tracking and safe autonomous navigation.

1.4 Motion Forecasting

One of the critical activities to be performed for safe and proactive navigation by an autonomous vehicle is motion forecasting. It needs to predict the future trajectories of the dynamic objects, such as, vehicles, pedestrians, and cyclists, around the vehicle. Precise motion forecasting enables the autonomous vehicle to anticipate the movement of such objects and, therefore, come to far better and wiser decisions to prevent a potential collision. This capability is necessary to deal safely and smoothly with complex environments.

The challenges in motion forecasting in an autonomous vehicle have included prediction of dynamic behaviors, treatment with uncertainty, real-time processing, complex scenarios, and integration with perception and guidance. This needs to predict the various future movements of different objects with their respective behavior patterns. This would need an understanding of the intents and actions of other road users. By their very nature, the future motions of dynamic objects are not certain. Uncertainty should be handled by motion forecasting systems and lead to trustworthy predictions. For real-time performance, motion forecasting has to be performed in time to allow immediate and precise responses to dynamic changes within the environment. The system should be perfected in offer-

ing enough performances within diverse and challenging environments, such as urban areas with thick traffic and intersections, where road users' behavior cannot be predicted. It needs to be combined with perception and guidance in a vehicle for a coherent and complete view of the environment in which motion forecasting is performed to enable safe navigation.

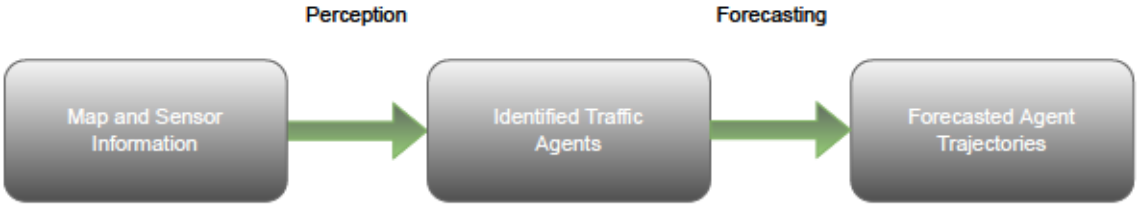

Figure 1.3: Motion forecasting flow diagram.

The new methods for motion forecasting in autonomous driving are based on a wide area: from purely kinematic models, to purely machine learning models, and to the hybrid models. Kinematic models utilise physical equations in order to tell what future states of objects are supposed to be regarding their current state. Although simple and fast, they may not capture complex behaviors. Then there are machine learning methods such as RNNs and LSTM networks that learn from the data the evolving patterns for trajectory forecasting. While these will easily model complex behavior, this involves huge amounts of data and significant computational resources. A good prediction accuracy and robustness is generally achieved by the combination of kinematic and machine learning models. These approaches must be combined by using sophisticated algorithms and extensive tuning. A simple flow diagram of motion forecasting steps is given in Figure 1.3.

While these have been huge steps forward, the presently published motion forecasting techniques cannot deal with either the diversity or the complexity of real environments, provide the level of accuracy necessary to guarantee safety, or interface well with perception and guidance systems. It is expected that new ap-

proaches are proposed that can show an ability to overcome such challenges in the context of the provision of reliable motion forecasting for autonomous vehicles. Deep learning and AI techniques offer promising solutions for these challenges. In this regard, neural networks and complex algorithms can be developed into more accurate and efficient motion forecasting systems. AI-driven techniques learn from vast amounts of data, handle uncertainty, and process predictions in real time, that will help us build trustworthy motion forecasting systems for the safe and efficient operation of autonomous vehicles.

1.5 Research Objectives

Following the background set forth in this introductory chapter, this thesis focuses, first and foremost, on the development and application of advanced AI-based techniques towards improving perception and motion forecasting for an autonomous vehicle. Having presented very realistic challenges for autonomous driving, this research work employs deep learning and AI methodology to construct safe, reliable, and efficient approaches for developing autonomous vehicles.

A key goal of this thesis is to develop appropriate approaches for the accurate segmentation of LiDAR point clouds in real-time using various neural network architectures. First would be leveraging CNNs, and spherical projection techniques with synthetically generated data from advanced autonomous driving simulators like CARLA. The existing real-world data is limited in variety that this creates several challenging scenarios. Synthetic dataset were generated to support training and model performance improvement, since adequate and diverse real-world data is not available. Merging it with real-world datasets, such as KITTI, will therefore help in improving the training and validation of these models for better generalization when applied across different environments and scenarios. The second

approach employs Vision Transformerss (ViTs) on processing LiDAR data for the segmentation. LiDAR segmentation should be as accurate as picking up surrounding information, such as understanding and identification of objects and obstacles, including all kinds of road features. By utilising these approaches, involving CNNs with synthetic data augmentation and ViTs, we aim to improve the segmentation process in scenarios involving changing environmental conditions, such as varying lighting, weather, or the presence of moving objects. The expected outcome in this case is that the better performance of LiDAR segmentation, hence giving the detailed and accurate view of the surroundings of the vehicle and performing well under different conditions; therefore, reliable and safe driving under autonomy is guaranteed.

An additional objective of this work is to further develop a novel multi-object tracking framework that exploits transformer-based architectures together with sensor fusion for enhanced accuracy in dynamic environments. This approach will combine 2D and 3D object detection by data merging between LiDAR and cameras for better comprehension of traffic agents' movements. By fusing the sensor data in real time, the system can accurately estimate the position and velocity of moving agents. The incorporation of transformer networks into the tracking framework allows both the local interactions and global context to be modeled with sophisticated attention mechanisms. It should be effective in multi-object tracking in complex scenarios, especially those taking place while driving in urban environments.

The final objective of the thesis is, making an accurate estimate of the trajectories of dynamic objects around the vehicle. Motion forecasting plays an important role here, where the idea is to forecast the subsequent motions of other traffic agents so that the autonomous vehicle can take correct decisions toward collision avoidance. This objective will be facilitated through transformer networks and

other advanced forms of deep learning models through analysis for object motion forecasting. These state-of-the-art AI techniques will, therefore, be instrumental in enabling the motion forecasting system to manage such complexity and uncertainty regarding dynamic environments. Consequently, this is expected to yield a robust motion forecasting system capable of enabling an autonomous vehicle to predict the behavior of other road users and safely react on time toward proactive navigation.

In summary, the specific aims of this thesis are to:

- develop methods for real-time, accurate segmentation of LiDAR point cloud with different neural network model variants to enhance the environmental perception of the vehicle,
- integrate synthetic and real-world data to increase the reliability of the perception system in autonomous vehicles,
- develop a multi-object tracking framework using transformer networks and sensor fusion to enhance the accuracy of tracking in dynamic environments to enhance vehicle's understanding and the capability of traffic tracking,
- forecast, with high accuracy, trajectories of dynamic objects around the vehicle using transformer networks and deep learning sophisticated models, enhancing proactive navigation.

1.6 Outline and Contributions

This thesis is organized into several chapters that contribute to the overall goal of autonomous driving technologies by proposing new approaches in perception, tracking, and motion forecasting.

Remark 1: Research Questions

1. Can synthetic data obtained from simulators be used as a supplement for the real dataset in order to improve the generalization and robustness of a deep learning model in a self-driving car?
2. Can techniques be developed to enhance the accuracy and speed of LiDAR point cloud segmentation in real time utilising different neural network architectures for improving the vehicle environmental perception?
3. How can sensor fusion and transformer networks be adapted to enhance multi-object tracking in dynamic environments?
4. Can transformer networks and state-of-the-art deep learning models be applied to predict with high accuracy the trajectory of dynamic objects so navigation of an autonomous vehicle can also be proactive?

- **Chapter 2** covers a summary of the necessaary background concepts for the current research, including the topics of AI, Machine Learning (ML), LiDAR segmentation, object tracking, and motion forecasting.
- **Chapter 3** presents a case for the integration of synthetic data with real-world data for LiDAR segmentation, showing how synthetic data improves both the performance and generalization of models.
- **Chapter 4** describes a novel approach in LiDAR segmentation by using the Vision Transformer architecture and underlines benefits of this architecture for enhancing the accuracy of segmentation.
- **Chapter 5** introduces a sensor-fusion and transformer-based, object track-

ing framework to shed light on concrete ways in which transformers can be adapted for tracking in dynamical environments.

- **Chapter 6** provides an overview of the development of Intention Refined Motion Transformer for Motion Forecasting (IRMTR), hence effectively predicting multi-agent trajectories with high accuracy.

- **Chapter 7** summarizes the main contributions of the thesis and discusses any implications from these findings; directions for future research are also indicated.

All the chapters are arranged in logical order to ensure that the most relevant aspects of autonomous driving, such as LiDAR segmentation, object tracking, and motion forecasting, with their key challenges, can be covered.

1.7 Published and Submitted Manuscripts

Conferences

- Inan, B. A., Rondao, D., and Aouf, N. (2023a). Enhancing lidar point cloud segmentation with synthetic data. In *2023 31st Mediterranean Conference on Control and Automation (MED)*, pages 370–375. IEEE.

- Inan, B. A., Rondao, D., and Aouf, N. (2023b). Harnessing vision transformers for lidar point cloud segmentation. In *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1–6. IEEE.

- Inan, B. A., Aouf, N. Transformers for Enhanced Multi-Object Tracking with Sensor Fusion. In *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)*.

Journals

- Inan, B. A., Aouf, N. IRMTR: Intention Refined Motion Transformer for Motion Forecasting. *Advanced Intelligent Systems – Wiley Online Library (under review)*

CHAPTER 2

Theoretical Background and Tools

This chapter introduces concepts and techniques lying at the rear of research into self-driving vehicles, based on deep learning methods for perception, object tracking, and motion forecasting. Starting from covering topics within the area of artificial intelligence, presenting an overview of machine learning, deep learning, and the architectures of neural networks that are the basis to enable advanced perception capabilities. k-means Clustering and Gaussian Mixture Model (GMM) are also necessary mathematical models that will also be discussed in that chapter, where their roles in data representation and segmentation are demonstrated. The chapter also gives an overview of CNNs and transformers, pointing out in detail their application in handling LiDAR point cloud data for scene understanding and motion forecasting. The background presented in this chapter forms a complete toolbox for the derivation of methods and systems that are presented in the subsequent chapters.

2.1 Artificial Intelligence

AI basically refers to the intelligence of machines programmed to feel, think, and learn like human beings. The term was coined for the very first time in 1956 by John McCarthy during the Dartmouth Conference, and with that began AI as a separate field of study. AI is a broad term appearing to describe a wide variety of technologies and methodologies developed to enable machines to carry out tasks which generally require human intelligence, including reasoning, learning, problem-solving, perception, and language understanding.

While the concept of AI has long been established in the circles of science, technology, engineering, and mathematics, it has truly only been of great media concern as of late. Although AI is commonly used to refer, quite broadly and sometimes imprecisely, to machines that think, the core objective of AI is to create machines that can perform tasks as well as—or perhaps even better than—humans. Probably the first, and best-known benchmark for AI, is the Turing Test Turing (1950), proposed by Alan Turing in 1950. The test measures the extent to which a machine can demonstrate intelligent behaviour indistinguishable from that of a human. According to, Russell & Norvig (2016), the area of AI has developed to such an extent that there are considered to be some specific key areas in which a computer should have capability; otherwise, it could not be considered as intelligent. Such capabilities are:

- Natural Language Processing: To communicate fluently in human languages, such as in English.
- Knowledge Representation: The way in which it would organize and store the information that it gathered.

- Automated Reasoning: The employment of stored knowledge to answer questions and infer new facts.
- Machine Learning: In order to learn how to deal with situations completely new, to recognize patterns, and predict.

These are key areas put together that allow a computer to emulate intelligent behaviour. AI can be applied to a wide range of areas in diverse fields, including:

- Healthcare: AI is applied for diagnostics of diseases, creating treatment plans, and even for drug discovery.
- Finance: AI algorithms are used for fraud detection, risk management, and automated trading.
- Transportation: AI helps in developing autonomous vehicles, traffic management systems, and route optimization.
- Manufacturing: AI optimizes supply chains, predictive maintenance, and quality control.

2.1.1 Machine Learning

While AI in general is a broad area of science where desirable and often complex actions of a machine can be performed without explicit programming, ML is a more narrow sub-field of AI that concerns the development of algorithms which allow computers to learn from, and make predictions or decisions based on, data. Unlike traditional computer programming, in which the computer is given explicit instructions, the ML system learns and improves its performance over time as it discovers patterns in the data. Big data have become the trend of time as

large datasets are easily available. The more availability of information and abundant than ever before requires the evolution of techniques to handle this bulk of data. Basically, ML is divided into two types: *supervised learning* and *unsupervised learning* Murphy (2012). Supervised learning is a type of learning that seeks to perceive any pattern or relationship between inputs and outputs based on pairs of data bounded by labels. In contrast, unsupervised learning aims to identify important patterns in the input data of interest; without labeled data or correctly targeted outputs are needed. Such ML, by its nature, is less structured. Problems are more naturally framed in the context of supervised learning, despite unsupervised learning being much broader in its applicability. For example, humans tend to learn by observing their surroundings.

Whatever the machine learning approach, for computers that learn by observation, computer vision plays the role of impacting perception about an object, just like in humans. Combined with robotics, and the above-mentioned four areas, it represents one of the six fundamental areas making up most of AI Russell & Norvig (2016). The capability of capturing detailed information about the environment in a clear-cut manner, together with small and inexpensive off-the-shelf cameras, makes computer vision one of the most preferred ways of input data acquisition for robotics applications. Besides, computer vision is also the nearest machine equivalent of human object perception, while the field of robotics is among the key disciplines which extensively applies AI effectively. The versatility of robotics platforms, combined with the accessibility of small, compact, and cost-effective cameras, makes them the preferred choice for computer vision applications. Their ease of capturing raw, detailed information about the environment makes the cameras an invaluable input channel for robotics applications.

In general, ML consists of two main stages: First comes the feature extraction stage, where discriminative and informative subsets are derived from the raw

data. LiDAR segmentation and motion forecasting depend on the data for the establishment of multi-level associations between the data points and their associated contexts in meaningful ways by considerable feature extraction from the raw input data. For instance, LiDAR segmentation is supposed to utilise key features out of the point cloud data for perceiving the vehicle environment. For the motion forecasting task, features are extracted from dynamic object data to forecast further trajectories. In the second step, a proper model is selected that can process the extracted features into desired results. These models usually undergo some optimization processes, which converge at minimum prediction errors by granting assurance of accuracy and reliability in the results.

2.1.2 *k-means* Clustering

The term "*k-means*"was coined by Macqueen (1967). The *k-means* clustering is an unsupervised ML algorithm that segments unlabelled data into non-hierarchical distinct clusters. As earlier discussed, an unsupervised machine learning is a type of machine learning where one trains a computer to analyze unclassified, unlabelled data without any interference from a human being. In this case, the machine sorts out the data in classes, according to similarities, patterns and differences without being previously trained. The *k-means* algorithm is primarily a method of unsupervised learning that groups the data under consideration into a predefined number of clusters in such a way that within-cluster items resemble most to each other than objects in other clusters. In principle, it is one of the methods of grouping items considering their similarities and differences.

The *k-means* algorithm assigns each data point to one of the k clusters dependent on the distance from the centroid of each cluster. The process initiates with the random selection of initial centroids. Then, each data point gets allocated to

the closest centroid of a cluster. Subsequently, after making the assignment of all points, the centroids get recomputed with a view on the points contained in each cluster. This iterative process keeps running until the clusters stabilize and further changes become insignificant. In our analysis, we assume that the number of clusters, k, is predetermined, and each point is assigned to one of these clusters.

In some situations, the optimal number of clusters, k, is not clearly defined and it needs to be inferred. The algorithm works best when the data is well-separated; if there is a considerable overlap between points of different classes, it performs terribly. the *k-means* clustering, though widely used due to its fast speed and good performance on the grouping of data points compared with other clustering methods, does not return explicit information about the quality of the clusters. The cluster results will be different for different subjective initial centroid placements. Besides, *k-means* is sensitive to noise and may get stuck in local minima.

2.1.3 Gaussian Mixture Model (GMM) Clustering

A Gaussian Mixture Model is a machine learning model used for predicting the probability of a certain data point being associated with some cluster. GMM finds its application in unsupervised learning as a soft clustering method that determines the likelihood of data points to be from different clusters. A GMM is composed of several Gaussian distributions, each represented by $k \in \{1, \ldots, K\}$, where K represents the number of clusters in the dataset. Each Gaussian distribution k within the mixture is therefore parameterized by:

- The mean, μ, for the center of each Gaussian component, around which the highest point density of the data points is situated.
- The variance, Σ, is a concept showing dispersion of data with respect to the mean. A small variance implies that there are clusters of data packed close

to the mean while a large variance enforces large clusters.

- The weights, π, play an important role in a GMM, with weights telling us something about the number of data points actually belonging to each Gaussian component.

The weights here indicate the relative importance or prominence of each cluster in the overall mixture. Higher weights reflect the fact that, for the given Gaussian, a larger portion of the data corresponds to that particular type of distribution, making this part of the model more important. This combination of parameters, such as mean, variance, and weight, equips GMMs with great flexibility for modeling data. By changing this set of parameters, a GMM is able to approximate the distribution of a broad range of data-from compact to large-scale, even overlapping.

In fact, one of the most powerful abilities of a GMM, indeed maybe its core is determining the probability that each point comes from each cluster.

It does this through a process known as 'soft clustering', as opposed to other hard clustering methods such as *k-means*. Instead of assigning each data point definitely to one cluster as it did in soft clustering, GMM returns the probabilities that express the likelihood of a certain data example belonging to each Gaussian component.

2.1.4 Deep Learning

Deep learning is one of the sub-fields of ML where a model learns directly from raw data inputs, without necessarily going through the conventional manual feature extraction. It is interchangeably used with the term *Artificial Neural Networks (ANNs)*, as the first models were inspired by functions of the human brain

Goodfellow et al. (2016). The fundamental ANN model is the *multi-layer perceptron* (MLP; Bishop & Nasrabadi (2006). This model produces a vector of outputs $\mathbf{y} = [y_1 \dots y_K]^T$ from an input vector $\mathbf{x} = [x_1 \dots x_D]^T$, by computing M linear combinations of the input elements and passing them through a nonlinear activation function h:

$$a_j = \sum_{i=1}^{D} W_{ji}^{(1)} x_i + b_j^{(1)}, \quad j = 1, \dots, M \tag{2.1}$$

$$z_j = h(a_j), \tag{2.2}$$

where $W_{j,i}$ and b_j are the learnable parameters of the network, weights, and biases respectively. The superscript (1) indicates the first layer of the MLP. For the simplest case of an ANN with only a single *hidden layer*, the output activations are directly obtained from z_j through another linear combination:

$$a_k = \sum_{j=1}^{M} W_{kj}^{(2)} z_j + b_k^{(2)}, \quad k = 1, \dots, K, \tag{2.3}$$

Depending on the type of response, the output activations can be either fed through another nonlinear activation function or taken as identity as $y_k = a_k$ (see Figure 2.1). The advancement of optimisation techniques and improvement with high computing capability have allowed the training of deeper models that contain more hidden layers in reasonable time frames using consumer-grade *Graphics Processing Unit (GPU)*. This progress spurred a second wave of research in the area in the late 2000s, and somewhat promoted the term "deep" learning—often referred to as *Deep Neural Networks (DNNs)*, as synonymous with state-of-the-art neural networks.

Similar to traditional ML, the training of a DNN typically performed by minimizing a scalar loss function $f(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of learnable parameters

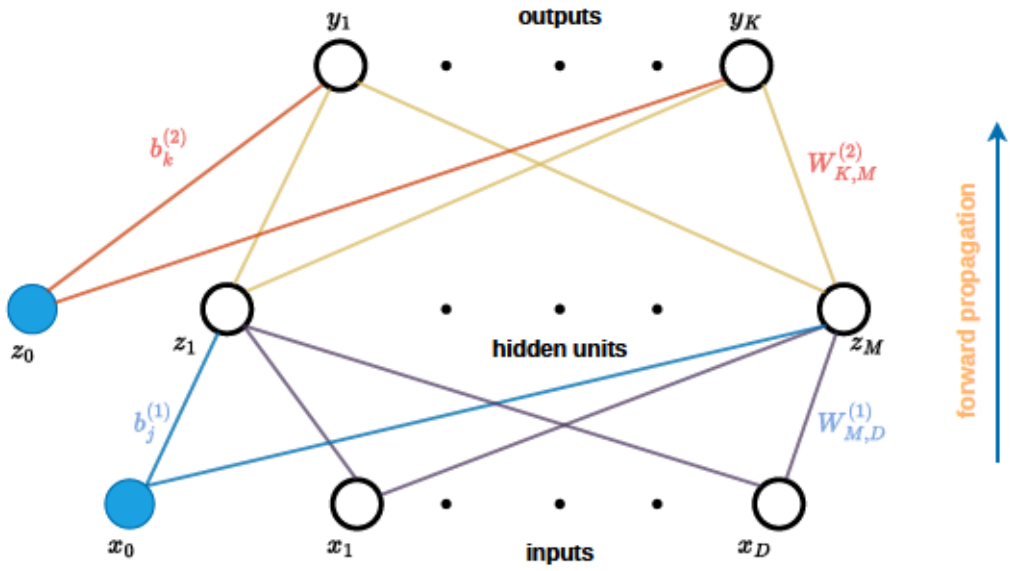

Figure 2.1: Diagram depicts a two-layer artificial neural network ANN. The open nodes indicate input (x), hidden (z), and output (y) units, with connections between these units representing weights (W). Biases (b) are shown as connections from additional closed nodes (x_0, z_0).

of the network, and dependency on the inputs and outputs is explicitly indicated. Owing to DNN being inherently nonlinear (as pointed out by Eq. 2.2), the optimisation problem of the optimal $\boldsymbol{\theta}$ requires iterative methods that search for critical points in f using gradient information. The simplest method is to take a small step in the $\boldsymbol{\theta}$-space in the direction of the negative gradient,which is termed as *gradient descent* Bishop & Nasrabadi (2006):

$$\boldsymbol{\theta}^{(K+1)} = \boldsymbol{\theta}^{(K)} - \alpha_K \nabla_{\boldsymbol{\theta}} f, \tag{2.4}$$

where α_k is the learning rate at time step $\tau = \tau_k$. One of the main merits of Equation (2.4) is that it does not require the computation of the Hessian.

A single forward pass through the DNN provides the necessary conditions to

calculate the gradient of the loss with respect to the output, i.e. $\nabla_{\boldsymbol{y}} f$. The gradient of f with respect to the weights in each layer, which is essential for gradient descent optimization as described in Equation (2.4), can be obtained by successively chaining the local gradients of each layer in reverse order until the target layer is reached. This process is known as *backpropagation* Rumelhart et al. (1986). When a unit j in one layer feeds into k units in the next, the local gradient at j is given by:

$$\frac{\partial f}{\partial a_j} = \sum_k \frac{\partial f}{\partial a_k} \frac{\partial a_k}{\partial a_j}. \tag{2.5}$$

Once the gradient values of the output units are calculated via a forward pass, recursive back-propagation can efficiently calculate the gradients for all hidden layers for any DNN model.

Since the gradient-based parameter optimisation of the cost for large datasets is usually bound by memory, a minibatch $\mathcal{B} = \{x^{(1)}, \ldots, x^{(m)}\}$ of m inputs may be sampled from the training data set at each step and utilised instead. This approach transforms the process into a *Stochastic Gradient Descent (SGD)*, where the gradient is approximated as:

$$\nabla_{\boldsymbol{\theta}} f \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} f(x^{(i)}, y^{(i)}, \boldsymbol{\theta}). \tag{2.6}$$

An epoch is said to be complete when the SGD algorithm processes all the mini-batches that together comprise the complete dataset of inputs. Once this iterative process is done, it ensures that every single input in the dataset has played its role in contributing to the model's learning process. Practically, DNN training normally requires more than one epoch to adequately converge the parameters in a network to optimal performance. Each epoch gives an opportunity for the DNN to incrementally refine the inner representations, by which its accuracy improves

and generalizes with time.

Such a linear output as in Equation (2.3) allows a neural network to learn regression problems. However, the appeal of deep learning today has been large, mostly because of outstanding performance for *classification* tasks where outputs are discrete and usually mutually exclusive, and selections come out of a large set of choices. While the sigmoid function (Figure 2.2, top right) is applied to model outputs that have a Bernoulli distribution (e.g., coin toss, dog vs. cat classification, or pedestrian vs. vehicle classification), the *softmax* function defined as

$$\text{softmax}(a)_k := \frac{\exp(a_k)}{\sum_l \exp(a_l)}, \tag{2.7}$$

is used to model categorical, or generalized Bernoulli, distributions. This becomes handy for things like distinguishing different types of vehicles, or distinguishing between different environmental contexts in autonomous driving.

2.1.5 Activation Functions

Early ANN used sigmoid or hyperbolic tangent activation functions (Figure 2.2, top right and top left, respectively) Goodfellow et al. (2016). The sigmoid functions have a tendency to saturate across most of their range, being sensitive only in an area close to zero. This complicates the process of learning because of vanishing gradients during backpropagation. Although they once enjoyed widespread usage in early networks, their use in practice is now generally not recommended except in special cases (§ 2.1.9). Hyperbolic tangent functions are easier to train but still suffer from the lack of large gradient when activation inputs are even close to zero.

It has one special property: it returns zero for negative inputs and it linearly responds to non-negative inputs, (Figure 2.2, bottom left). This makes it very easy to optimize. Its advantages include: it does sparse activation, where about 50%

of the hidden units in a randomly initialized ANN produce non-zero output. This sparsity prevents overfitting and hence improves the generalization capabilities of the model for unseen data (see § 2.1.13). Moreover, ReLU allows better gradient propagation than sigmoid and hyperbolic tangent; in fact, it is not suffering from the problem of saturation for both small and big input values.

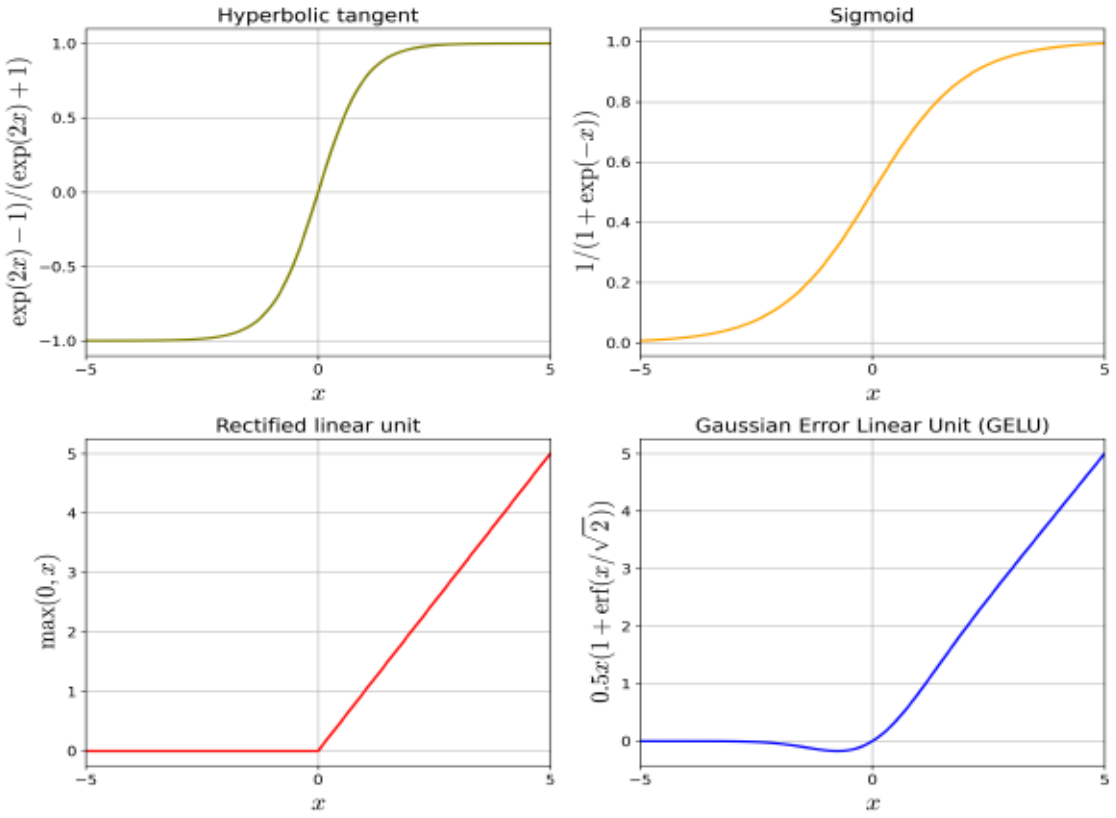

Figure 2.2: Nonlinear activation functions for artificial neural network ANN. Top Left: $h(x) = tanh(x)$. Top Right: $h(x) = sigm(x)$. Bottom Left: $h(x) = \text{relu}(x)$. Bottom Right: Gaussian Error Linear Unit $h(x) = \text{gelu}(x)$.

A extremely popular and modern choice of activation function is the Rectified Linear Unit (ReLU), which is defined as:

$$\text{relu}(x) := max(0, x). \tag{2.8}$$

Remark 2.7: ReLU and Leaky ReLU comparison.

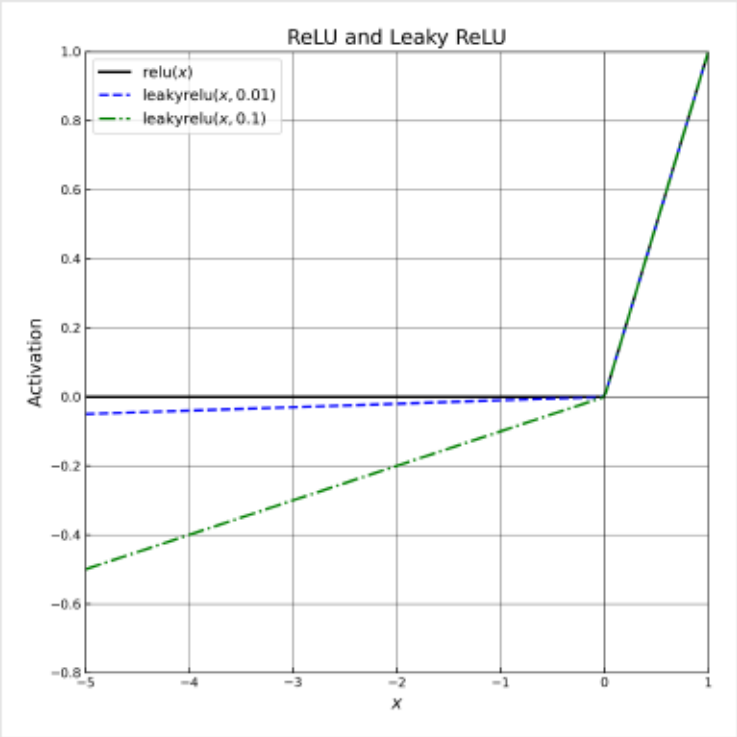

Figure 2.3: Comparison between the rectified linear unit (ReLU) activation function and the leaky ReLU activation function. The latter is plotted for two distinct values of η, the parameter defining the slope of the response when $x < 0$.

Despite its advantages, the ReLU activation can run into specific problems during backpropagation, ReLU units can become inactive regardless of the input, a phenomenon known as the dying ReLU problem. This occurs when the network learns a large negative bias, the activation outputs zero (see Figure 2.2, bottom left), effectively halting the backward gradient flow. To tackle this issue, several generalisations of the ReLU function have been proposed. For example, the leaky ReLU function (Maas et al. (2013)) allows a small gradient when the unit is not active:

Remark 2.7: Dying ReLU Problem

$$\text{leakyrelu}(x, \eta) := \begin{cases} x & \text{if } x > 0, \\ \eta x & \text{otherwise.} \end{cases} \tag{2.9}$$

The function is illustrated in Figure 2.4. Other variations, such as the parametric ReLU, treat η as a learnable parameter (He et al. (2015)).

Apart from the above-mentioned activation functions, the Gaussian Error Linear Unit or GELU really gained momentum in the last few years in transformer models and other sophisticated neural architectures. The GELU may be defined mathematically by the following function:

$$\text{GELU}(x) = 0.5x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{2.10}$$

The GELU activation function in Equation (2.10) combines the best of both worlds without compromising between linear and nonlinear activation functions; it offers smoother transitions and better performance in many situations. This special property of making learning much effective whereby the model may grasp interesting patterns in data much better, finds wide application in state of the art models such as BERT and, GPT due to these properties which greatly impact the overall efficacy and efficiency of such networks.

2.1.6 Optimisation

Section 2.1.4 introduced SGD as the "default" learning algorithm for deep neural networks. While simple, the learning process using SGD is often slow Goodfellow et al. (2016). To accelerate the optimisation, it is possible to inject some momentum into SGD. The impact of using momentum in gradient descent adds a friction term for two purposes: first velocity decides the direction of movement through

the parameter space, and second velocity combines with the gradient to decide the speed. The parameter update approach is defined as follows (see Equations 2.4 and 2.6):

$$v^{(\kappa+1)} = \gamma v^{(\kappa)} - \alpha \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} f\left(x^{(i)}, y^{(i)}, \theta\right) \right), \tag{2.11}$$

$$\theta^{(\kappa+1)} = \theta^{(\kappa)} + v^{(\kappa+1)}, \tag{2.12}$$

Equation 2.11shows that the SGD momentum retains the influence of past updates because the moving average does not forget them, where $\gamma \in [0, 1]$ decides the speed at which the influence of older gradients decays. Momentum helps in overcoming non-essential valleys where the Hessian matrix is badly conditioned, such as when the objective function's topology varies greatly in directions orthogonal to the gradient path toward a minimum. This variability would impact regular SGD much more.

However, momentum does not solve the problem of selecting the most critical hyperparameter. On the opposite, it actually introduces an additional parameter. Another approach has been developing optimization methods that adapt learning rates of model parameters. The RMSProp algorithm, introduced by Geoffrey Hinton in 2012[1], where each parameter in θ is updated by scaling their learning rates inversely proportional to the accumulated squared gradients over past steps. The accumulation is an exponentially decaying average: the exacerbation of the distant past values is minimal so that convergence can be fast when the algorithm finds a convex bowl Goodfellow et al. (2016). The update is given by:

$$g_{\text{acc}}^{(\kappa+1)} = \gamma g_{\text{acc}}^{(\kappa)} + (1 - \gamma) g^{(\kappa+1)} \odot g^{(\kappa+1)}, \tag{2.13}$$

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

$$\Delta\theta^{(\kappa+1)} = -\frac{\alpha}{\sqrt{g_{\text{acc}}^{(\kappa+1)} + \epsilon}} \odot g^{(\kappa+1)}, \tag{2.14}$$

$$\theta^{(\kappa+1)} = \theta^{(\kappa)} + \Delta\theta^{(\kappa+1)}, \tag{2.15}$$

where $\odot$ denotes the element-wise product, ϵ is a small constant to stabilize division by potentially small numbers, and g is the gradient.

The Adam algorithm Kingma (2014) extends this by adding a term resembling momentum based on the non-squared gradient. It also adds bias correction estimations to both gradient-dependent terms. The full update is summarized as follows:

$$g_{\text{1st}}^{(\kappa+1)} = \gamma_1 g_{\text{1st}}^{(\kappa)} + (1-\gamma_1) g^{(\kappa+1)}, \tag{2.16}$$

$$g_{\text{2nd}}^{(\kappa+1)} = \gamma_2 g_{\text{2nd}}^{(\kappa)} + (1-\gamma_2) g^{(\kappa+1)} \odot g^{(\kappa+1)}, \tag{2.17}$$

$$\hat{g}_{\text{1st}}^{(\kappa+1)} = \frac{g_{\text{1st}}^{(\kappa+1)}}{1-\gamma_1^{\kappa+1}}, \tag{2.18}$$

$$\hat{g}_{\text{2nd}}^{(\kappa+1)} = \frac{g_{\text{2nd}}^{(\kappa+1)}}{1-\gamma_2^{\kappa+1}}, \tag{2.19}$$

$$\Delta\theta^{(\kappa+1)} = -\frac{\alpha \hat{g}_{\text{1st}}^{(\kappa+1)}}{\sqrt{\hat{g}_{\text{2nd}}^{(\kappa+1)} + \epsilon}}, \tag{2.20}$$

where the divisions and square roots are applied element-wise and the parameter update is identical to Equation 2.15. Note that in the bias update steps, the decay parameters γ_1 and γ_2 are exponentiated by the current iteration value κ. With this way, Adam features increased robustness to the choice of hyperparameters relative to SGD with momentum or RMSProp.

2.1.7 Convolutional Neural Networks

Traditional ML techniques often require a pre-processing stage in order to feed an input into the training algorithm. These inputs are rarely raw data but rather the outcome of some feature detection and extraction algorithm that produces a set of observations $\mathbf{X} = \{x^{(1)}, \ldots, x^{(N)}\}$. This step is important because it ensures that selected features represent the observed reality properly, and this adequacy can vary significantly depending on the application. For example, once an algorithm becomes specialized in detecting features in LiDAR point clouds, then it should be robust to various variabilities that it might actually encounter in real-world autonomous driving, such as changing weather and different road surfaces.

On the other hand, deep neural network systems have advantages in image-based applications due to their end-to-end training capability, optimal feature description can be learned directly from data in an unsupervised manner, bypassing the need for handcrafting the features. This is very helpful in segmenting LiDAR and thus allows DNNs to be trained in various driving environments. For example, to capture the intrinsic nonlinearities between sensor data and segmented objects in the environment, segmenting methods based on DNN can be very useful for a self-driving car. The hidden layer structure,as highlighted earlier, inherent to the Multi-layer Perceptron (MLP) is the linear combination between every possible input and output; each layer is thus called a Fully Connected (FC) layer. Training an FC layer on image data would incur a high computational cost due to the number of pixels involved. Convolutional neural networks are better suited for such data representation, as suggested by Le Cun et al. (1989).

A convolution operation slides a 2D kernel $\mathbf{K}$ (often a small square matrix with odd dimensions) across a 2D input image $\mathbf{I}^{\text{in}}$ to produce a 2D output, $\mathbf{I}^{\text{out}}$, according to the operation:

$$I^{\text{out}}_{i,j} = (\mathbf{I}^{\text{in}} * \mathbf{K})_{i,j} = \sum_u \sum_v I^{\text{in}}_{i+u,j+v} K_{u,v}, \tag{2.21}$$

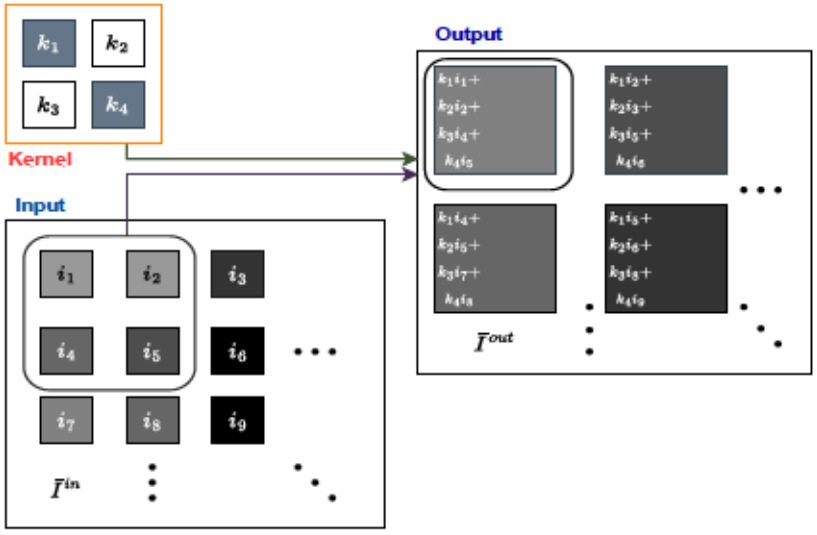

Figure 2.4: Example of a two-dimensional convolution operation. A single-channel input image $\mathbf{I}^{\text{in}}$ is processed using a 2×2 kernel $\mathbf{K}$ to generate a single-channel output image $\mathbf{I}^{\text{out}}$. This illustration is only limited to the top-left 3×3 sub-matrix of $\mathbf{I}^{\text{in}}$, with each input pixel labeled as $\{i_1, \ldots, i_9\}$. Correspondingly, the kernel elements are indicated by $\{k_1, \ldots, k_4\}$.

where index notation has been used. The immediate advantage of convolutional layers is the sparsity of learnable parameters, which reduces significantly compared with FC layers, reducing the memory requirements. This is inherently owing to a parameter-sharing trait, meaning the single set of parameters is learned regardless of the input location. This kernel slides over the entire image, detecting features in a localised manner, while creating a feature map representative of the location and intensity of these detected features. CNNs often come in the form of feature extraction front-ends, successively reducing spatial information while generating additional feature maps. These feature maps can be interpreted as images

having multiple channels or, in other words, spatial dimensions.

The convolution can be extended from a 2D kernel to a 4D tensor, $\mathbf{K}$, with dimensions given by:

$$\dim\mathbf{K} = C_{\text{in}} \times F \times F \times C_{\text{out}}, \tag{2.22}$$

where C_{in} is the number of input channels, C_{out} is the desired number of output channels, and it is assumed that the kernel is spatially square with dimensions $F \times F$. Spatial reduction can be done by applying a pooling operation after the convolution, where the feature map of spatial dimensions W_{in} is subdivided into bins to generate an output of dimension:

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - Q}{Q} \right\rfloor + 1, \tag{2.23}$$

where Q is the size of the pooling window. Common pooling operations are the maximum or average value of each bin. In practice, most recent DNNs have abandoned the pooling layers in favor of increasing the stride, S, of the convolution, i.e., the number of rows and columns of the input that are skipped while sliding the kernel. In such cases,the size of the spatial output can be controlled by:

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - F + 2P}{S} \right\rfloor + 1, \tag{2.24}$$

where P is the spatial padding applied to the input.

2.1.8 Transfer Learning

DNNs that form complex models usually require very long training times and large datasets. An alternative is the technique of transfer learning based on the assumption that some of the causes underlying the output of a task are also underlying the

output of another task. On CNNs, it may be assumed that most of the learned kernels converge to recognize generalised visual features. In practical applications, it has been empirically observed concerning the field of application that first layers' kernels of CNNs are optimized for broad features that are wide across the domain, such as corners and edges, while the different kernels in the final layers will focus on more task-specific shapes Zeiler & Fergus (2014). This mostly consists of taking a pre-trained CNN architecture-whose initial layers have gained expertise from a large, generalized dataset-and fine-tuning those final layers, or even adding additional new ones, on small-sized domain-specific datasets.

Pre-training on the vision-related task typically makes use of the ImageNet dataset, containing over 14 million labelled images across more than 20,000 categories, thanks to crowdsourcing. Major developments in CNN architecture design have been spurred by participation in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), targeted at developing an ML pipeline for correctly classifying images into 1,000 distinct ImageNet categories. The important breakthrough for DNN came when Krizhevsky et al. (2012) won the ILSVRC 2012 with a top-5 classification error of 15.3%, more than 11 percentage points better than the runner-up, using a variant of Le Cun et al. (1989) CNN trained on a GPU, which hugely reduced computation time.Some of the new CNN designs have annually competed in this challenge, and as a result, there have been substantial advances every year, but also there has been an introduction of innovative breakthroughs. For instance, VGG Simonyan & Zisserman (2014) is notable for its use of very small (3x3) convolution filters and its depth, which significantly improved the accuracy over the earlier models. GoogLeNet Szegedy et al. (2015) was also distinguished by its deep architecture but with parallel layers, where multi-scale features could be extracted. Similarly, ResNet He et al. (2016) took it even further by introducing residual connections that allowed training of networks even deeper. Many of

these modern DNN architectures have been released open-source, with publicly available pre-trained ImageNet weights, an arrangement that has substantially accelerated the pace in this field and allowed for the wide adoption of these models as CNN front-ends on many applications.

2.1.9 Recurrent Neural Networks

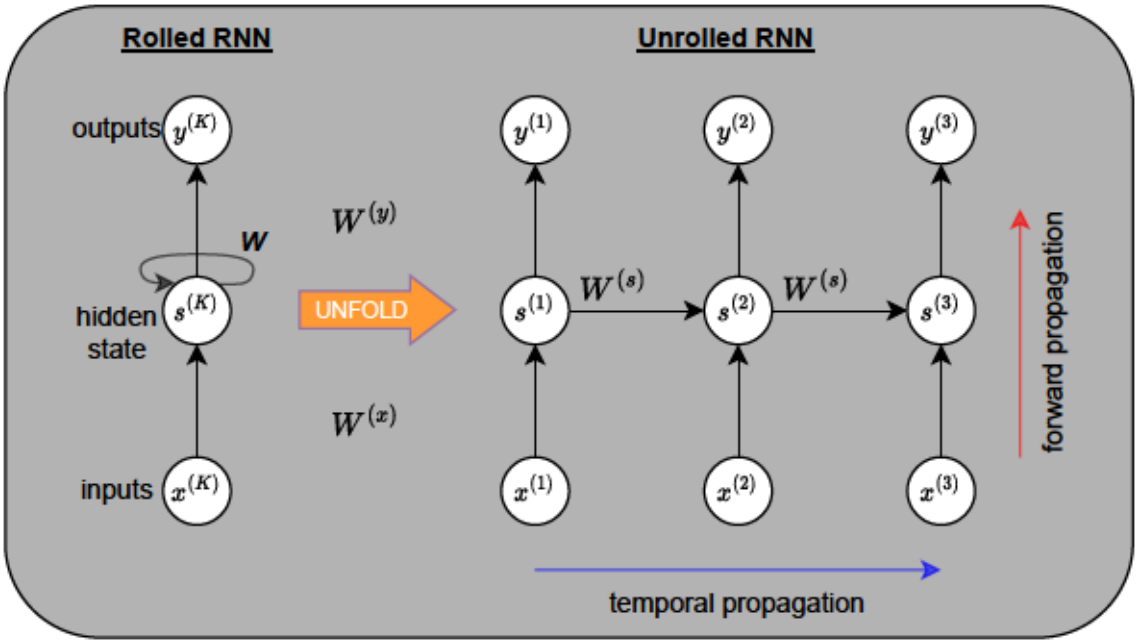

Figure 2.5: Depiction of a basic two-layer RNN and the unfolding architecture. In this type of network, parameter training at time-step $\tau = \tau_k$ is influenced not only by the current input $x^{(k)}$ but also by the previous output $s^{(k-1)}$. The unfolded/unrolled representation forms a chain-like structure, with gradient computation occurring via forward propagation along a temporal axis. This simple diagram suggests the repetition of the same network architecture. Network elements are illustrated using vector notation with individual nodes.

The architectures introduced so far treat each input separately at each forward pass. *RNNs*, on the other hand, are designed capture temporal dependencies be-

tween inputs across a series of time steps. These networks contain loops, that allow information to be conveyed from one step to the next, and hence have a recursive or chain-like structure when unfolded in time (see Figure 2.5). The basic RNN architecture includes an additional set of parameters to weigh the influence of prior outputs Rumelhart et al. (1986):

$$a_j^{(k+1)} = \sum_{i=1}^{D} W_{ji}^{(x)} x_i + \sum_{k=1}^{M} W_{jk}^{(s)} s_k^k, \quad j = 1, \ldots, M \tag{2.25}$$

$$s_j^{(k+1)} = h\left(a_j^{(k+1)}\right), \tag{2.26}$$

where the bias term has been omitted for simplicity (cf. Eq. 2.1). Such a network architecture is quite powerful in modelling sequential data, e.g., image captioning, Natural Language Processing (NLP) and time-series data such as vehicle trajectories. The recurrent weights $W^{(s)}$ are optimized by Backpropagation Through Time (BPTT), an algorithm similar to standard back-propagation of acquiring $W^{(x)}$, yet with errors accumulated at each time step due to shared parameters temporally. The gradient w.r.t the recurrent weights of the model presented in Figure 2.5 is computed by the following equation:

$$\frac{\partial f}{\partial W^{(s)}} = \sum_{k=1}^{T-1} \sum_{j=1}^{M} \frac{\partial f_k}{\partial y_k} \frac{\partial y_k}{\partial s_k} \left(\prod_{k=j+1}^{K} \frac{\partial s_k}{\partial s_{k-1}} \right) \frac{\partial s_j}{\partial W^{(s)}}, \tag{2.27}$$

where T denotes the total number of time steps and f_k depicts the cost function value at time $\tau = \tau_k$.

The key issue in traditional RNNs is due to the term $\prod_{k=j+1}^{k} \partial s_k / \partial s_{k-1}$ in Equation 2.27, as repeated matrix multiplications that may lead to vanishing or exploding gradients. For the vanishing gradients, it results in the cessation of learning by the network, whereas for the exploding gradients, there is numerical instability. The other problem of RNN involves learning long-term dependencies Goodfellow et al. (2016). To alleviate these problems, several improvements have been done,

the most noticeable being the *long short-term memory* unit Hochreiter & Schmidhuber (1997). These LSTM units consist of an input gate to regulate the effective running and regulation of the cell state; thus, it acts like an "information highway", letting only the selected information flow through and therefore alleviating the problems within traditional RNNs.

2.1.10 Attention Mechanism

Attention mechanisms in neural networks are inspired by cognitive attention.These mechanisms aim at improving model performance through dynamic focus on the most relevant parts of the input data. Attention mechanism was first introduced to improve encoder-decoder models applied in machine translation. It enables the decoder to utilise those parts of the input sequence that are most relevant by taking a weighted combination of all encoded input vectors, giving higher weights to the vectors that are most relevant.

Attention was introduced by Bahdanau et al. (2014) to tackle the bottleneck of the fixed-length encoding vectors problem. However, this fixed-length context vector, in some traditional encoder-decoder models, restricts the decoder from accessing input information, especially when the sequences are long or complex. These sequences, although they are more complex, are forced to be represented in the same dimensionality as shorter or simpler sequences, restrict the performance of the model. Readers are encouraged to refer to Bahdanau et al. (2014) for more detailed information on the attention mechanism.

In a typical encoder-decoder architecture, the fixed-length context vector becomes a bottleneck for sequence-to-sequence learning since it overemphasizes the latter part of the sequence. Since tokens or words are fed into the model one after another, the context vector ends up focusing much on the last parts of the input

sequence. he typical encoder-decoder model works well only when sequences are short, often less than 20 tokens.

Attention mechanisms overcome this limitation by introducing attention vectors, serving as new context vectors at each decoder time step, considering all tokens of the input sequence and selectively focusing on the important ones unlike standard models. Each context vector is computed as a weighted sum of "annotations." These annotations carry information about each input token in relation to its surrounding tokens, while the weights are determined by the alignment between the current annotation and the previous decoder hidden state. The weights over the annotations serve to select which of the input tokens should be given more priority in computing each of the context vectors.

In case of time series data, a Gaussian is used to re-weight the data, giving greater weight to time steps closer in time. This has the effect of dampening the noise levels appreciably, which is suitable for time series data. However, in other kinds of data, such as images or text, proximity alone cannot be the only criterion for weighting. For example, in text, words a few words apart from each other in a sentence are not necessarily more alike than words that are far apart from each other. Instead, another weighting scheme that carries more context is desirable.

Self-attention mechanisms can be used to obtain similarity measures between embeddings without any bias toward proximity. This makes usage of the dot product between embeddings to decide about their similarities, enabling the model to attend to the most relevant parts regardless of their positional distance in the input data. This technique, known as self-attention, was popularized by Vaswani et al. (2017).

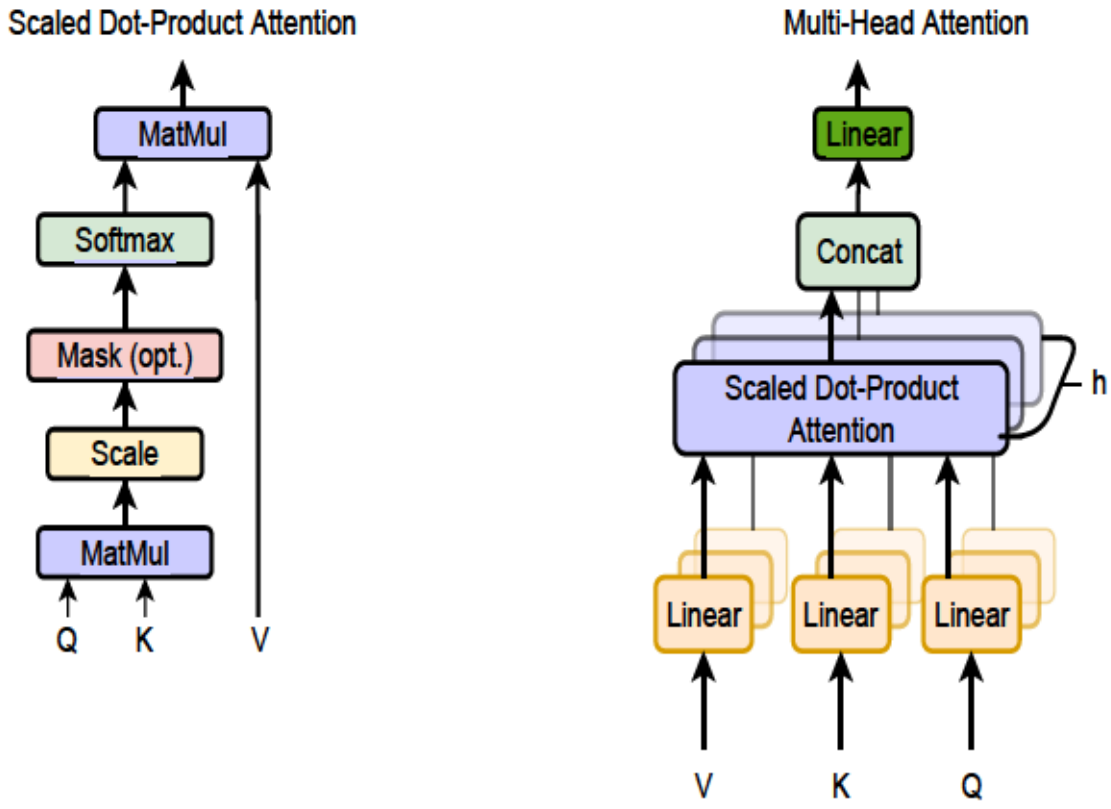

Figure 2.6: Left: Scaled Dot-Product Attention diagram. Right: Multi-Head Attention comprises multiple attention mechanisms working in parallel Vaswani et al. (2017).

Scaled Dot-Product Attention

The transformer architecture takes advantage of a mechanism called self-attention, more precisely scaled dot-product attention, to process its inputs Vaswani et al. (2017), can be seen in Figure 2.6. This mechanism takes as input queries and keys of dimension d_k, and values of dimension d_v. The dot products of the query with all keys are computed, scaled by $\sqrt{d_k}$, and then passed through a softmax function to determine the weights on the values. This will allow the model to focus on different parts of the input sequence when generating an output.

The self-attention function is computed on a set of queries simultaneously;

these are packed together into a matrix Q. Keys and values are packed similarly into matrices K and V, respectively. The output matrix is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.28}$$

Two prevailing functions that are used as attention are additive attention and dot-product attention, which has the theoretical same complexity as dot-product attention, but dot-product attention is more efficient in practice since it has been implemented through highly optimized matrix multiplication code.

Smaller values of d_k, give similar results between additive and dot-product attention mechanisms. However, for larger values of d_k, the dot-product attention without scaling can have large magnitude values that make the softmax function produce very small gradients. In order to avoid this problem, the dot products are scaled by $\frac{1}{\sqrt{d_k}}$,helping reduce the vanishing gradient problem Vaswani et al. (2017).

$$\text{SoftMax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.29}$$

Here, i is the current element and the summation over index j acts as a normalization term. In this formula, Q is the set of queries, K is the set of keys, and V is for a set of values. Those matrices can be considered as abstractions extracted from an input vector multiplied by a learned matrix during training. The scaling factor $\sqrt{d_k}$ is applied for mitigating the vanishing gradient problem. So when the dimensionality d_k is large, the dot products grow really large and applying the softmax on these large values may lead to extremely small gradients, which are undesirable. Finally, the result is a matrix of associations of the words or image patches, supplying some context to each embedding.

Multi-Head Self-Attention

Vaswani et al. (2017) demonstrated that instead of using one attention mechanism with d_{model}-dimensional queries, keys and values, it is beneficial to linearly project queries, keys and values h times with different learned linear projections to d_q, d_k, and d_v dimensions, respectively. The attention function for these projected versions of queries, keys and values is computed in parallel, resulting in d_v-dimensional output values. These outputs are then concatenated and projected once again, which provides the final values given as shown in Figure 2.6.

Multi-head attention allows the model to attend information regarding different representation subspaces across multiple positions, which a single attention head would not allow because of constraints enforced by averaging.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2.30}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.31}$$

In this context, the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

2.1.11 Transformers

The transformer is a deep neural network component to capture informative representations of sequences or sets of data points Vaswani et al. (2017). It has an autoregressive model, and has driven rapid progress during recent years in natural language processing Devlin et al. (2018), computer vision Dosovitskiy et al. (2020), and spatio-temporal modeling Bi et al. (2022). The architecture of the transformer can be seen in Figure 2.7 in detail.

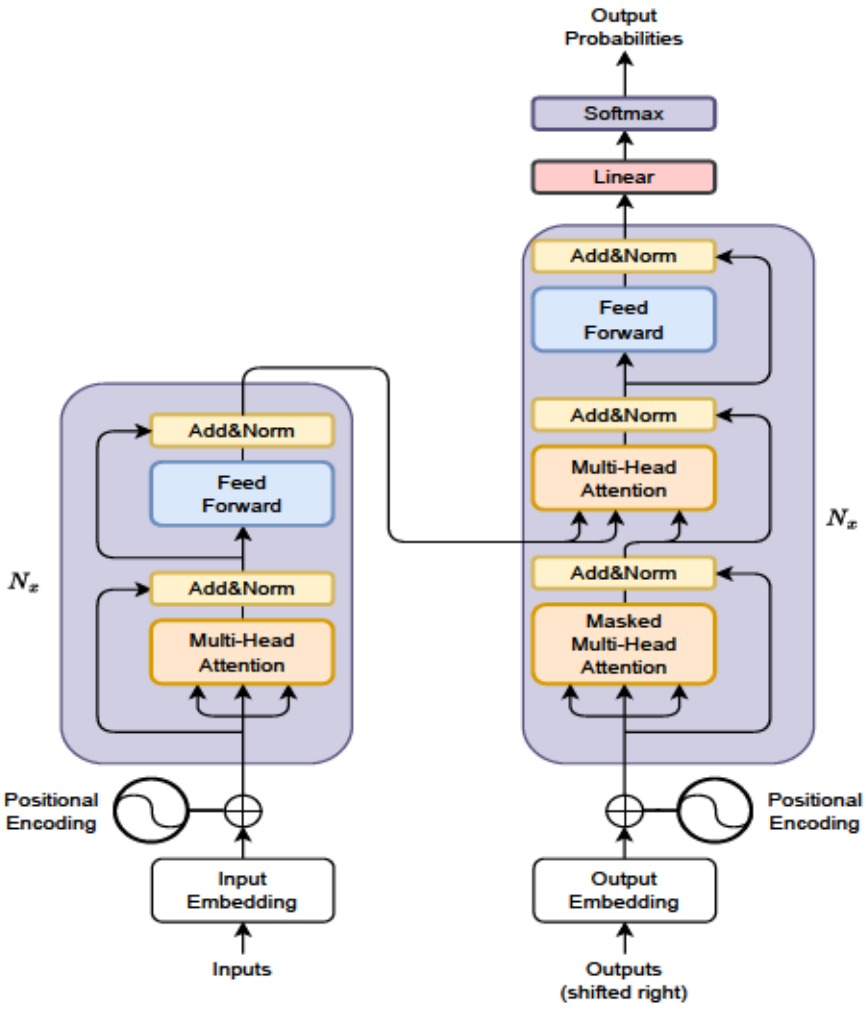

Figure 2.7: The transformer model architecture, On the left is a single encoder block, stacked N times. On the right is a single decoder block, stacked N times, Vaswani et al. (2017).

From Seq2Seq to Transformers: A Paradigm Shift

Before delving into the transformer architecture, the Sequence to Sequence (Seq2Seq) model in Sutskever et al. (2014) must be mentioned. Seq2Seq, using a two multi-layered LSTM-based encoder and decoder, maps the source input sequence into a fixed-dimensional hidden representation first and decodes it into an output sequence. This model's both input and output structure is similar to the transformer.

Seq2Seq model in machine translation processes an input sentence to develop a hidden representation that would be utilised by a decoder for the generation of an output sentence. The decoder dynamically creates the sentence, in which a token is fed into the decoder in order to get the first word, that would afterwards be fed into the decoder until the generation of the end sentence token. This idea can already be generalized for motion forecasting where the encoder operates on the past motion data and a decoder predicts future trajectories.

One critical problem with Seq2Seq is the bottleneck from condensing all the information into a single hidden representation vector, h. This is particularly disastrous for long or complex sequences. Previous approaches attempted to address this using the vanilla mechanism of attention. And the idea was pretty much similar to having the decoder access all input tokens at each time step. This featured an improved model performance and resolved some issues related to the loss of gradients, but it still maintained the sequential nature of RNN architectures; hence, the models could not be scaled well. Eventually, these very limitations contributed to the emergence of the development of the transformer Vaswani et al. (2017), a purely feed-forward architecture, which has been an evident way out of scalability issues associated with RNNs.

Embeddings

The very first thing one needs to do, before going deep into the specifics of the transformer architecture, is to introduce some basic concepts, starting with embedding. In ML generally, an embedding is a way to represent data in vector form of any dimensions. That is because a neural network cannot process it if there is a raw sequence of characters. We embed information into a numeric vector using embeddings, which then becomes apt for use by deep learning models. This section introduces embedding and lays the basic understanding that will be helpful

in further understanding the transformer architecture.

In order to make use of a transformer, data needs first to be transformed into a set or sequence of N tokens, each of dimension D. These tokens are gathered together into a matrix $X^{(0)}$ of dimensions $D \times N$. Two concrete examples are as follows:

- A representation for a text can be performed as a sequence of words or sub-words, where each word/sub-word has a unique vector associated with it.
- For a given image, it can be segmented into multiple patches; each is subsequently mapped to a vector.

The embeddings may either be fixed or learned together with the model parameters. For instance, vectors for words may be optimised; image patches may be embedded using a learned linear transform.

A sequence of tokens is a generic representation for input data. Different kinds of data can be "tokenized," so that transformers apply universally and without requiring specialized architectures for each type of data, such as CNNs for images or RNNs for sequences. That flexibility means transformers can mix different modalities of data simply by tokenizing them into one set of tokens.

Like other sequence processing models, this model utilises learned embeddings to convert both the input tokens and output tokens to vectors of dimension d_{model}. It also uses a standard linear transformation followed by a softmax function to convert the output of the decoder into probabilities of next token. Similar to Press & Wolf (2016), in this model, the weight matrix is shared between these two embedding layers and pre-softmax linear transformation. Also, embedding layers multiply these weights by $\sqrt{d_{model}}$ for proper scaling.

The transformer takes the input data $X^{(0)}$ and outputs a representation of the sequence in the form of another matrix $X^{(M)}$, with dimensions $D \times N$. Each slice

$x_n^{(M)}$ is the representation of the sequence at the position of token n. These representations are being used in a variety of applications: predicting the next token given all the previous ones, global classification of the whole sequence, sequence-to-sequence learning, image-to-image prediction problems, among others. Here, M denotes the number of layers in a transformer.

In summary, embeddings are a preparatory step that allows transformers through the process to have various types of data inputted into them since they would all have been transformed into vectors of equal dimensions. This transformation thus easily allows transformers to apply deep learning methods on sequences of data points by making it yet another versatile tool in ML.

Positional Encoding

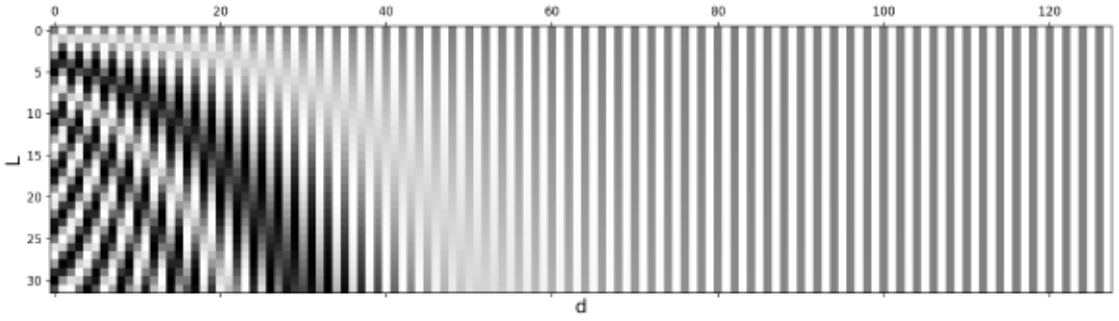

Figure 2.8: Sinusoidal positional encoding with $L = 32$ and $d = 128$ where the values range from -1 (black) to 1 (white), with 0 represented in gray.

In order to realize the full potential of transformers in sequence-based tasks, positional information about the token being processed should be introduced one way or another. Traditional architectures innately capture such orders of sequences, which is crucial in most tasks where the order of elements matters. Whatever permutation operation one performs on the input provided to the transformer

architecture, the model itself will not know or care; hence, necessitating the development of positional embeddings to retain sequential information.

To handle this problem, positional encodings are added to the input embeddings at the bottoms of the encoder and decoder stacks. These encodings have the same dimension, d_{model}, as the embeddings, allowing their summation. While there are many positional encoding strategies, sinusoidal positional encodings created by Vaswani et al. (2017) are particularly effective. This is illustrated in Figure 2.8.

This technique embeds positions through a look-up of sine and cosine functions of different frequencies. The positional encoding for a position pos and dimension i is defined as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.32}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.33}$$

Here, pos represents the position, and i denotes the dimension. Each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. This design choice allows the model to easily learn to attend to relative positions, thus aiding in easy learning process for sequences of variable lengths.

Another popular strategy is to utilise learned the positional encodings. Instead of being fixed, it is initialized at the start and learned with the rest of the model parameters. Both fixed and learned positional encodings work well, as several experiments show performance that is quite close to one another.

The most important advantage of the sinusoidal encodings is that they allow the model to extrapolate to sequence lengths not seen during training. This is very desirable in models which should generalize well across different sequence

lengths.

However, in various applications, this positional information has to be infused differently. For example, in vision transformers Dosovitskiy et al. (2020) adds the positional embedding to patch embeddings so it may have proper processing over image patches.

In the transformer model, since positional encodings have been added, it is capable of capturing the order of sequences for its processing, hence stronger in handling many of these tasks which are sequence-based.

Encoder and Decoder Stacks

The transformer encoder consists of multiple identical layers, with the original model having six such layers, each of which changes input sequences into continuous representations that are abstract, capturing learned information from the whole sequence. This is facilitated by two main constituent sub-modules:

1. A multi-head attention mechanism.

2. A fully connected feed-forward network.

Each of these sub-layers is wrapped with a residual connection He et al. (2016), followed by layer normalization Ba et al. (2016). Specifically, the output of each sub-layer is computed as $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ represents the function executed by the sub-layer itself.

Feed-Forward Neural Network

The output from the normalized residual connections is further processed via a point-wise Feed Forward Network (FFN) that is important for further refinement. It is structured in a way that it is made up of two linear layers separated by a ReLU activation, mathematically represented as follows:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.34}$$

While the linear transformations are consistent for different positions, the parameters differfor different layers. This architecture can also be interpreted as two convolutions with a kernel size of 1. The dimensionality of input and output is $d_{\text{model}} = 512$, while the inner-layer dimensionality is $d_{\text{ff}} = 2048$.

After it has been processed, the output is once again combined with the pointwise FFN input, following up with another normalization step to fit and prepare the output adequately for the next stages.

Output of the Encoder

The last encoder layer outputs a sequence of vectors, each vector rich in their respective contextual information from the input. The vectors will go as input into the transformer decoder.

This elaborate encoding process informs the decoder to pay attention to certain parts of the input while decoding. Think of it as building a tower, whereby more encoder layers can be added on top. Each layer in this stack looks at, and learns in various ways, the different aspects of attention to add up to the knowledge base. This commencement not only begins to diversify the understanding but also increases the predictive capabilities of the transformer network.

2.1.12 The Transformer Decoder

The decoder is mainly used for generating the text sequence. Similar to the encoder, the decoder is also stacked with sub-layers, such as two multi-headed attention layers and one point-wise feed-forward layer; residual connections are added to these, followed by layer normalization after each sub-layer. These work

similarly to the layers in the encoder but perform somewhat different functions. The last layer in the decoder is a linear layer for classification, then the softmax function figures out the probabilities of different words.

The decoder was designed to decode information step by step to create an output. It operates in a fully autoregressive manner, starting with a given start token. It takes as input the previously generated outputs, along with the encoder's outputs, which are rich in attention information stemming from the initial input. It keeps decoding until it predicts such a token that signals the end of the creation of the output.

The entire process starts on the input line of the decoder, where the input initially flows through an embedding Layer in the same way as the encoder. This goes through the positional encoding layer right after the embedding layer. These positional embeddings are input to the first multi-head attention layer of the decoder; this is where the attention scores specific to the input of the decoder are calculated.

It consists of a stack of identical layers, six in the original transformer model. Each layer has three major subcomponents.

The mechanism of masked self-attention is rather similar to the self-attention mechanism used in the encoder; but has a significant difference in preventing the positions from attending to subsequent positions. This is to be sure that each word in the sequence is not influenced by future tokens. For example, it shouldn't take the "you" token into account when calculating the attention scores for "are", because "you" is a later position in the sequence. This masking function allows the model to depend only on known outputs at previously considered positions to make a prediction for a particular position.

The encoder and decoder parts interact in the second multi-headed attention layer of the decoder, where queries and keys come from the encoder outputs, while

values are from the first multi-headed attention layer of the decoder. The reason for this is to allow the decoder to align with the input of the encoder, underlining the most relevant parts. Further refinement of the output from this attention layer is done through a point-wise feed-forward layer. In this sublayer, the queries come from the previous decoder layer, whereas keys and values come from the output of the encoder. This way, every position in the decoder can attend to all positions in the input sequence, incorporating information from the encoder with that of the decoder.

Similar to the encoder, each decoder layer is also followed by a fully connected feed-forward network applied identically and independently at every position. The transformer model finishes processing the data in passing through a final linear layer acting as a classifier. It is as large as the number of total classes; the classes being the words that make the vocabulary. For instance, with the number of 1000 classes; then the classifier would give the output as an array of 1000 elements. This feeds into a softmax layer, which converts this output into probability scores ranging between 0 and 1. The highest probability score determines the most likely next word of the sequence.

Each sub-layer (masked self-attention, encoder-decoder attention, feed-forward network) is followed by normalization and includes a residual connection around it. The output of the final layer is transformed into a predicted sequence, usually via a linear layer plus softmax to produce probabilities over the vocabulary. This decoder output then becomes the new input in the growing list of inputs fed into the decoder as it continues decoding. This is done continuously until the model outputs a special token that signifies the completion of the process. It predicts the token with the highest probability and that will be labeled as the final class, normally determined by the end token.

To recall, the decoder is not limited to a single layer. It can be structured with

multiple layers, where every subsequent layer builds from an input of the encoder and previous layers. This layered architecture now allows the model to diversify its attention and extract variable attention patterns across its attention heads. The multi-layer approach enhances the model's predictive ability by building a much finer understanding of various combinations of attention.

2.1.13 Regularisation

The model capacity in an ordinary sense means the number of learnable parameters in deep neural networks, thus implicitly becoming a hyperparameter. While it is a fact that increasing the model capacity enhances the ability of a model to make correct predictions, high capacity can also cause *overfitting*: instead of learning from the train data, the network may memorize it, and generalizing to new, unseen test data will be poor. Overfitting generally happens if the number of the network's parameters is too large.

It is quite standard to test the performance of a model on a validation set from time to time during training in order to get an idea of the model generalization capability. Unlike the training sets, whatever data is being used for validation sets should not be used for training purposes. Instead, it is used to check how well the model is generalizing. If the training error is radically smaller than the validation error, that is a good indicator of overfitting. On the other hand, if there are a few parameters, high training and validation errors can also result in underfitting.

In order to handle overfitting, one can decrease the number of parameters or use early stopping, then halt the training process when the validation error starts to increase with respect to the training error. Besides that, one could make use of regularization techniques that introduce some type of noise in the learning process, in order to avoid overfitting. The regularization is an important procedure

unless the training dataset comprises millions of examples because it reduces the risk of overfitting and enhances the model's ability to generalize examples Goodfellow et al. (2016). This subsection gives a summary of some common regularization techniques for DNNs.

Dropout

Dropout is a technique in which a neural network randomly excludes the units by setting its output to zero Hinton et al. (2012). One can realize that this technique tries to emulate the ML scenario of bagging where K models are trained on K distinct subsets of the training data. For each mini-batch, a binary mask is randomly generated and applied to the hidden units of every layer with a given probability p. This fixed probability is used as a hyperparameter for the layer. Common values for p are 0.5 for FC layers, and between 0.1 and 0.2 for convolutional layers.

Weight Decay

Weight decay is actually a traditional and fundamental regularisation technique developed within ML, even before ANNs. This method involves including a term to the loss function that is proportional to the weight of each layer. A very common form of regularisation through weight decay is called L^2 regularisation, which incorporates the squared sum of all the weights, excluding, typically, the biases. This results in the following modification to the gradient of the i-th layer:

$$\nabla_{W_i} f(W^{(i)}, x, y) \leftarrow \nabla_{W_i} f(W^{(i)}, x, y) + \lambda W^{(i)}, \tag{2.35}$$

where λ is a hyperparameter typically chosen on a logarithmic scale, for example $\lambda \in \{10^{-6}, 10^{-5}, \ldots, 10^{-2}\}$.

Despite its long-standing history in ML, its alternative strategies are often utilised

jointly or as a replacement for weight decay, especially in modern CNNs.

Batch Normalisation

Batch normalization (Batchnorm), as introduced by Ioffe & Szegedy (2015), normalizes the inputs of a layer by calculating the mean μ_B and variance σ_B^2 for each mini-batch m as follows:

$$a'^{(0)} \leftarrow \frac{a^{(0)} - \mu_{Bm}}{\sqrt{\sigma_{Bm}^2 + \epsilon}} \tag{2.36}$$

Here, ϵ is a small constant for numerical stability. The outputs are then modified by a learnable scale and offset:

$$a''^{(0)} \leftarrow \gamma_{Bm} a'^{(0)} + \beta_{Bm} \tag{2.37}$$

During training, the mean and variance of the whole dataset are estimated by taking the moving average of each mini-batch's statistics $\{\mu_{Bm}, \sigma_{Bm}^2\}$. These values are then used to normalize the inputs at inference time. Batchnorm was originally proposed to improve DNN optimization, but it also introduces noise which can have a regularizing effect.

2.1.14 Image Augmentation

Image augmentation is one of the most powerful regularizations generating augmented additional data and guiding a DNN indirectly so that it pays its attention to essential features. In the case of a CNN, for example, its classifier can be instructed implicitly by rotating randomly an input image to make it invariant in terms of rotation—the object at a rotated location in LiDAR is the same object. This is achieved in a standard approach by randomly sampling a probability of applying various transformations on the input image at every iteration of training.

2.2 Semantic Segmentation of LiDAR Point Clouds

2.2.1 Point-Based Methods

LiDAR data are naturally represented in a 3D point cloud. Therefore, many approaches developed for 3D point cloud processing can be applied directly. An early work byQi, Su, Mo & Guibas (2017) presented a very powerful network named PointNet, which processes point clouds with raw data from a point cloud as input. It uses a symmetric function to sums the features of each point and extracts the most prominent feature for each dimension. While processing the features of individual points independently by a multi-layer perceptron, all point features are aggregated through max-pooling layers to generate a global representation. PointNet tackles the challenges brought about with permutation and rotation invariance of point clouds while providing a unified architecture applicable for various tasks such as object classification, part segmentation, and scene semantic parsing. The limitation of PointNet, however, is that the deeper features do not span wider spatial regions and thus fail to capture the local context and interaction among neighboring points. This issue was solved by Qi, Yi, Su & Guibas (2017), in which an enhanced version, known as PointNet++, was proposed; it consisted of a deep hierarchical network with sampling, grouping, and the core element of the PointNet backbone network. In particular, the improved scheme applies the farthest point sampling algorithm to select the most spatially distant points as the centers of local regions so that dimensionality reduction of the data does not lose the key geometric structures. Then, the grouping module builds the local regions, and the backbone network recursively learns the features of these regions.

KPConv Thomas et al. (2019) was a pioneering point convolution that acts directly on point clouds without requiring any intermediate representations. KP-

Conv's convolution weights are positioned in Euclidean space by kernel points and applied to the neighboring input points. It can also be extended to deformable convolutions, allowing the kernel points to adapt to the local geometry. Another approach, Tangent Convolutions Tatarchenko et al. (2018), which proposed another way for convolutional networks on 3D data operating directly on surface geometry and unstructured point clouds. In contrast, other methods do not operate in 3D space directly, projecting 3D coordinates into higher dimensional spaces first. For instance, LatticeNet Rosu et al. (2019) uses a sparse permutohedral lattice, PointNet is leveraged to describe the local geometry of point clouds and embedding them into the lattice for fast convolutions. LatticeNet is also proposing DeformSlice, a learned data-dependent interpolation projecting lattice features back onto the point cloud.

These techniques were, however, not specifically designed for the unique characteristics of LiDAR point clouds, which are typically sparse but contain a huge number of points. More recently, a paradigm of point-based techniques has popped up that adress the semantic segmentation of LiDAR point clouds. One such approach, Landrieu & Simonovsky (2018), which presented a deep learning-based framework that enabled semantic segmentation of large-scale point clouds, made up of millions of points, using superpoint graphs (SPGs). SPGs provide a compact yet informative representations of contextual relationships between object parts, that would be further processed by a graph convolutional network. Another work, Hu et al. (2020) proposed a simple and lightweight neural architecture for semantic segmentation of large-scale 3D point clouds. The proposed approach avoids complex point selection techniques and uses a random point-sampling approach. It also introduces a novel local feature aggregation module to expand the receptive field for each point. LiDAR segmentation methods are shown in Figure 2.10.

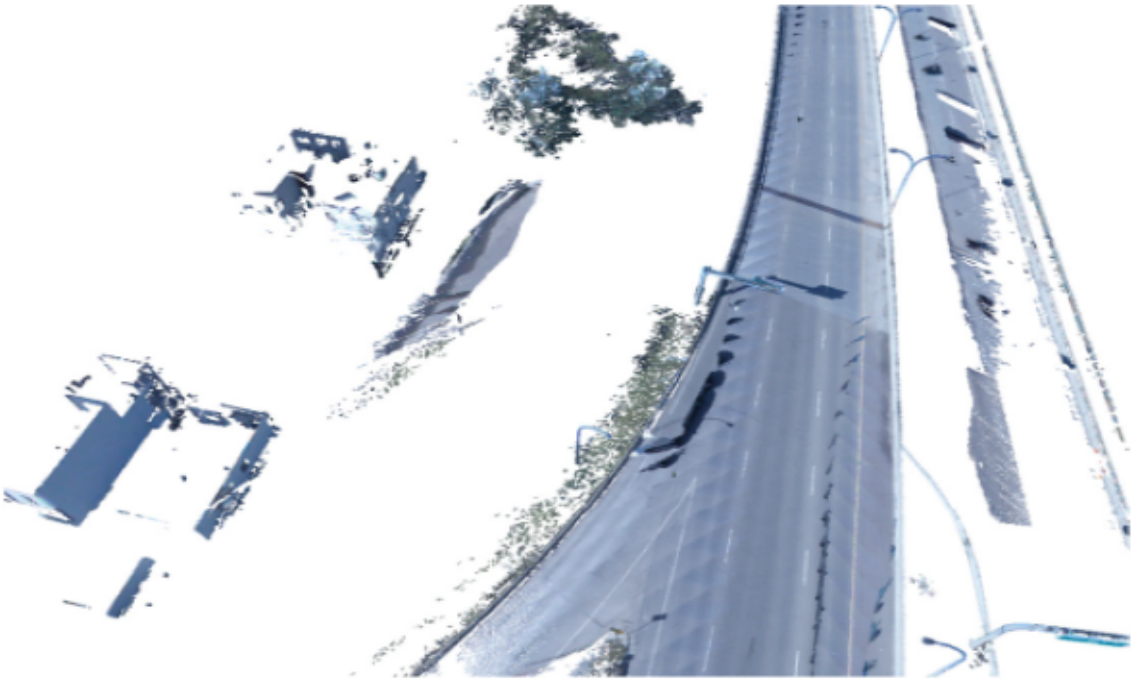

Figure 2.9: Nonuniformity of the point clouds. The visual is obtained from Zhang et al. (2023)

2.2.2 Voxel-Based Segmentation

While the results provided by point-based methods are not bad, most of them perform poorly, especially when utilised in LiDAR point clouds that have some special features like large-scale and sparsity. Most of these algorithms have not been optimized regarding those attributes; hence, they turn out very suboptimal. Most LiDAR point clouds are extremely sparse, with most of the space filled with 'nothing'. Voxelization is one of these ways of considering this unstructured data, where point clouds are mapped to a structured format. In the process of voxelization, objects are represented by voxels that are nearest to the points in the cloud. VoxNet Maturana & Scherer (2015) was the first attempt to perform voxelization, which transforms unstructured point clouds into forms of regular voxel grids and utilising 3D CNNs predict semantic labels for the occupied voxels by

use of standard convolutional operations. Although this approach overcomes the problem with unstructured point clouds, it still suffers from inefficiencies due to the sparsity of the voxel grid and high computational complexity associated with 3D CNNs. Non-uniformity of the point clouds illustrated in Figure 2.9

To further improve the efficiency in processing sparse voxels, Su et al. (2018) introduced SPLATNet, whose main interpolation is executed on the original point cloud so as to obtain a sparse voxel grid through a splat operation. It then applies convolution over the occupied voxels and, at the end, interpolates the output features using a slice operation back to the original point cloud. This approach is much more efficient since the indexing structure eliminates unnecessary computation for unoccupied voxels only to convolve the occupied voxels. The first work to tackle large-scale point clouds was that of Rosu et al. (2019) by proposing LatticeNet, which made use of PointNet as backbone. It is designed for fast convolution of sparse voxels with low computation overhead; it projects the features back onto the point cloud through a DeformSlice module. This has been very effective in handling very large-scale point clouds.

Tchapmi et al. (2017) introduced SEGCloud, an end-to-end semantic segmentation network that combines a 3D fully convolutional network with voxelization. The point cloud is first voxelized, and then 3D CNNs are applied to generate downsampled voxel labels. These voxel labels are transformed back into point labels using a trilinear interpolation layer. Interpolated scores are combined with point features through a 3D fully connected conditional random field and post-processing to acquire fine-grained semantic information.

However, even after the voxelization process, the LiDAR point cloud has inherent sparsity, which leads to sparse and discrete voxelized units. In this way, it introduces unnecessary computation cost. Moreover, segmenting point clouds into cubic regions is totally opposite to how a LiDAR sensor captures information.

PolarNet Zhang, Zhou, David, Yue, Xi, Gong & Foroosh (2020) addresses this issue by introducing a more effective grid representation in the online LiDAR point cloud semantic segmentation work. It proposes a new LiDAR-specific, nearest-neighbor-free segmentation method. Rather than using conventional spherical or bird's-eye-view projections, PolarNet adopts a polar bird's-eye-view representation that evenly distributes points across grid cells in a polar coordinate system.

Building on this, Cylinder3D Zhu et al. (2021) proposes a new framework for outdoor LiDAR segmentation that uses cylindrical partitioning and asymmetrical 3D convolution networks to explore 3D geometric patterns while preserving the intrinsic properties of outdoor point clouds. This model provides a backbone for other downstream tasks like point cloud semantic and panoptic segmentation or 3D detection. The next noticeable approach, Cheng et al. (2021), proposes an end-to-end encoder-decoder CNN for 3D LiDAR semantic segmentation. The contribution of this work is a new multi-branch attentive feature fusion module in the encoder and an adaptive feature selection module with re-weighting of the feature map in the decoder. It focuses on the existence of a number of challenges that almost all the current methods face, such as high computational complexity and loss of fine details in smaller instances.

Finally, Spherical Transformer Lai et al. (2023) proposes a method of spherical partitioning combined with a spherical transformer module to model the global and local geometric patterns of point clouds for an insightful understanding of data.

2.2.3 Projection-based Segmentation

Efficiency plays a great role in an autonomous driving context where real-time LiDAR point cloud processing is needed. While sparse convolution on 3D voxels

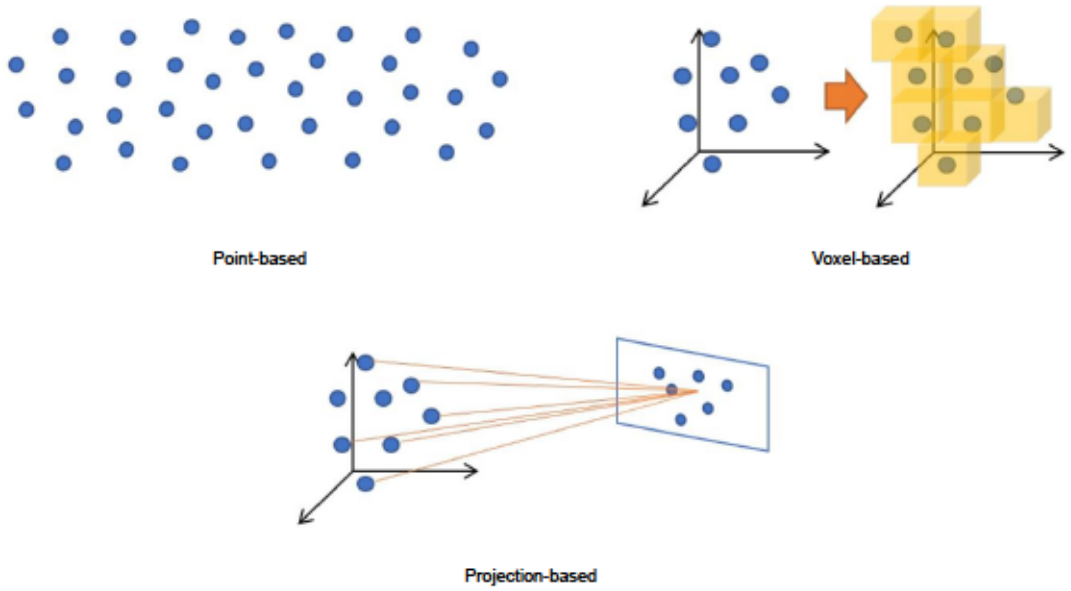

Figure 2.10: LiDAR segmentation methods Camuffo et al. (2022).

reduces the computational overhead to some extent, it is mostly too expensive for practical applications because of the demanding nature of 3D operations. A more efficient process would involve projecting the LiDAR point cloud onto a 2D projection map and carrying out semantic segmentation in this 2D representation. It can then be projected back in 3D.

RangeNet++ Milioto et al. (2019) leverages range images as an intermediate representation, integrating a CNN considering the rotating LiDAR sensor model. A new post-processing algorithm is introduced to reduce discretization errors and blurry outputs from CNNs, enhancing accuracy further. This work provided full semantic segmentation of LiDAR point clouds at sensor frame rate and showed, in practice, suitability for real-time applications quite effectively. Another notable approach, SalsaNext Cortinhal et al. (2020), builds upon SalsaNet Aksoy et al. (2020) with a new context module, residual dilated convolution stacks, and a pixel-shuffle layer while utilising Lovasz-Softmax loss for optimization. SalsaNext further enhances uncertainty estimation by using a Bayesian approach to compute

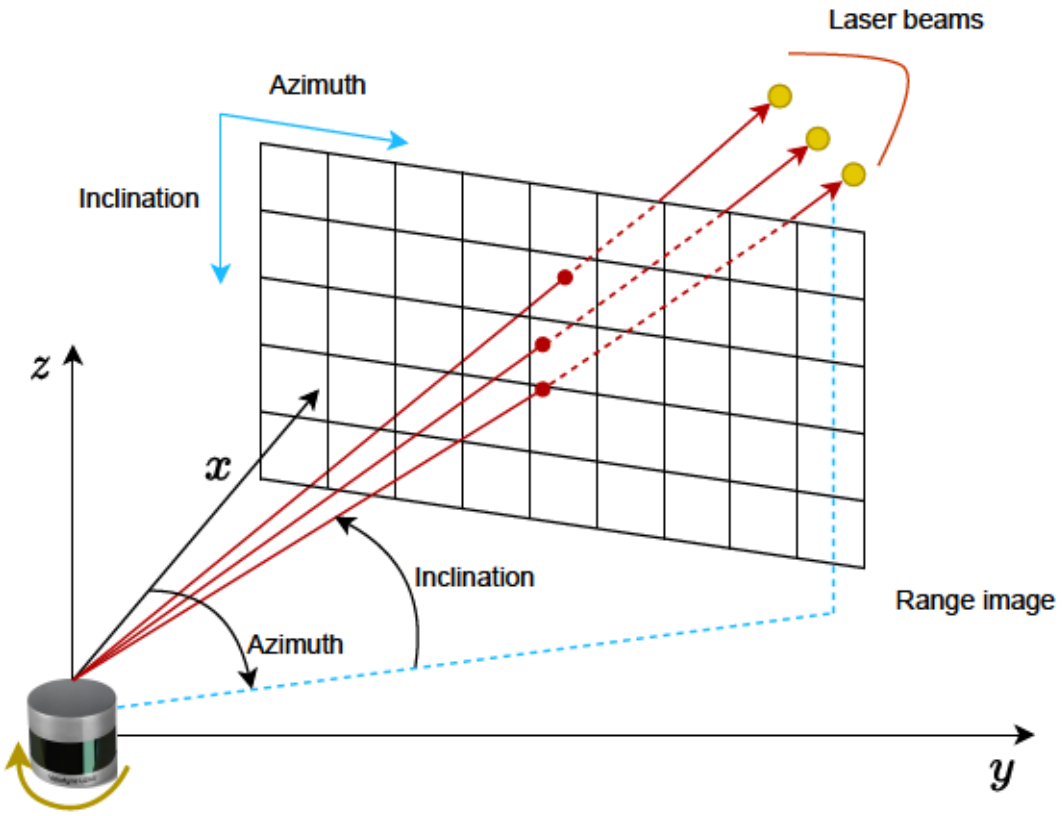

Figure 2.11: Process of the range image generation from a point cloud. The image is an adaptation from Fan et al. (2021)

the epistemic and aleatoric uncertainties for each point in the cloud, thus allowing for more robust semantic segmentation in real time. Another end-to-end pipeline, SqueezeSeg Wu et al. (2018), based on CNNs, generates point-wise label maps directly from transformed LiDAR point clouds. Next, this segmentation is further enhanced by using a Conditional Random Field (CRF) implemented as a recurrent layer. SqueezeSegV2 Wu et al. (2019) improves the robustness and accuracy of SqueezeSeg by modifying the model structure, training loss, and adding batch normalization, as well as additional input channels. This version also introduces a domain-adaptation training pipeline, enhancing performance on real-world data. Building on this, SqueezeSegV3 Xu, Wu, Wang, Zhan, Vajda, Keutzer

& Tomizuka (2020) proposes Spatially-Adaptive Convolution (SAC), which uses location-specific filters for different neighborhood regions in the projected images, allowing the network to effectively utilise the spatial features of point clouds. SAC is efficient and can generalize well, subsuming several prior methods as special cases. 3D-MiniNet Alonso et al. (2020) offers a fast and efficient approach for semantic segmentation of LiDAR point clouds by first learning a 2D representation through a novel projection that captures both local and global information from 3D data.

Then, this representation is used as input for a fully convolutional neural network to perform 2D semantic segmentation. Meanwhile, FPS-Net Xiao et al. (2021) leverages the uniqueness of projected image channels to improve the performance on segmentation. It uses an encoder-decoder architecture with an encoder that contains a residual dense block of multiple receptive fields, thus preserving detailed modality-specific information and learning the hierarchical features of the fused data effectively. KPRNet Kochanov et al. (2020) exploits image and point cloud segmentation techniques to improve LiDAR segmentation performance. It refines the CNN architecture in 2D projection-based methods and employs KPConv instead of classic post-processing for efficient and accurate generation of 3D labels. This approach has demonstrated better performance by incorporating learnable point-wise components. Finally, Rangeformer Kong et al. (2023) proposes the transformer architecture Vaswani et al. (2017) for LiDAR segmentation tasks, exploring the challenges and limitations of range view projections for semantic and panoptic segmentation. It points out three essentials affecting the performance of range view models: many-to-one mapping, semantic incoherence, and shape deformation that may provide guidelines for further studies on how to overcome such challenges.

2.3 Motion Forecasting

The capability to predict future motion concerning surrounding agents such as other vehicles, cyclists, and pedestrians is crucial in enabling autonomous vehicles to ensure safe and efficient navigation. It can be expected that proper forecasts of motion will keep the vehicle knowledgeable about traffic changes and correct its behavior accordingly. Coupled with spatial and temporal modeling techniques, motion forecasting facilitates an informed decision by a vehicle in real time through route optimization and the understanding of intents from other agents. This capability is particularly essential when passing through complicated urban environments where interactions between multiple agents are frequent and somewhat unpredictable.

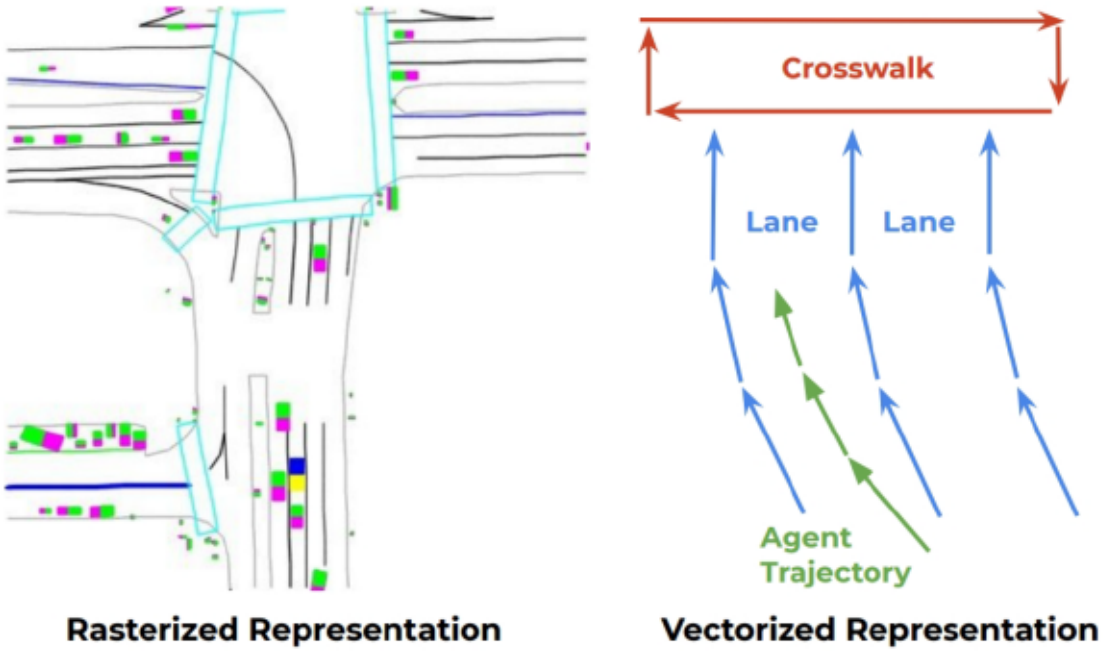

Figure 2.12: Rasterized grid representation (left) and vectorized method (right) for encoding high-definition maps and agent trajectories Gao et al. (2020).

2.3.1 Incorporating Map Information for Motion Forecasting

High Definition (HD) maps play a vital role in inferring driving context for motion forecasting. They are often adapted into specific formats, given the requirements of prediction models. Mainly, two types of maps are utilised: grid maps representing the environment and lane maps, providing information about lanes. These can be seen in Figure 2.12.

Grid Map Approach

The grid map approach, also termed as rasterized representation, divides the environment into a grid structure in which each cell includes full details about the surroundings. These are representative of the static road features such as boundaries, crosswalks, and intersections, plus dynamic objects like pedestrians and moving vehicles. It gives the machine learning models the input data in a structured, consistent way efficiently. The combination of real-time sensor data and HD map information conveys an effective snapshot of the current environment.

However, grid resolution is a trade-off between the level of detail and computational complexity. High-resolution grids capture finer details but, in return, require increased processing, while low-resolution grids are easier to use within computations since many details can be ignored, probably at the cost of losing critical features about the environment. Despite these challenges, the grid map format sees extensive use due to its scalability and ease to which prediction algorithms can be applied to provide a clear spatial overview of the driving scenario.

Lane Map Approach

In contrast, a lane map deals with just plain road lanes, their properties, and features a vectorized view of the road network. Lane maps capture some important

information such as the lane boundary, width, type (e.g., roundabouts, turning, or merging lanes), and the traffic flow direction while providing more structured notions of road layout to the autonomous systems. It is highly useful in complex driving scenarios, like crossroads or multi-lane roads, where lane relationships and the positioning of a vehicle are critical for accurate trajectory prediction.

Lane maps are created to be able to represent the hierarchical structure in the road; therefore, they can be useful in navigating complicated networks of roads. Since an autonomous system has already been trained with this detailed information, it could anticipate how vehicles will interact with the road infrastructure for more accurate trajectory predictions. This representation is particularly well-suited to those environments where strict lane adherence is required and often complements the broader view from grid maps.

2.4 Datasets

Datasets plays a central role in the evaluation of an algorithm. These are not simply pools of random, unrelated information elements but are a coherent set of ingredients which accurately represent the reality in which the method is to be validated and tested. Ideally, a dataset should sample information directly from the same reality, provided the circumstances allow it. Well-structured datasets enable the programmatic assessment of an algorithm's performance in comparison to others under identical conditions (benchmarking). With the widespread adoption of machine learning techniques, especially deep learning networks that feed directly on raw data as part of the training procedure, the importance of having access to meaningful and ample datasets has become increasingly crucial. Moreover, datasets should ideally be labeled, providing the ground truth from which the algorithm can simultaneously learn to classify or regress and be evaluated.

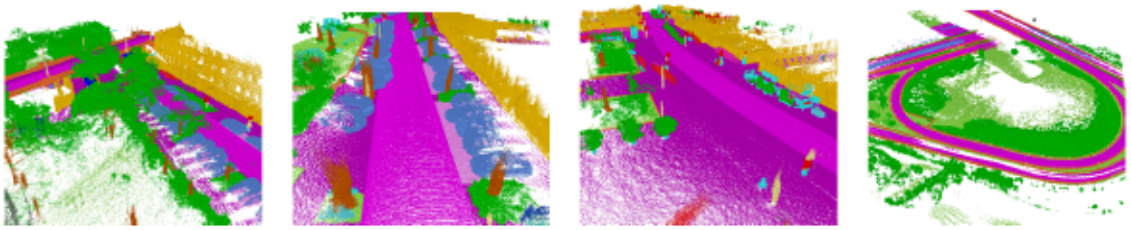

Figure 2.13: Visualisation for the SemanticKitti dataset. The visual is obtained from Behley et al. (2019)

In the context of this thesis, datasets are essential for the development and evaluation of autonomous driving systems. This work used open-sourced and our creation of synthetic datasets for model training and performance evaluation. For instance, segmentation and motion forecasting are critical tasks in autonomous driving, and the availability of labeled datasets for these purposes is varied. Although driving scenarios can differ from one region to another, the goal is to demonstrate the general efficiency and performance of the models. The use of simulation data offers several advantages, such as the ability to generate diverse and controlled scenarios, augment data efficiently, and safely test edge cases that might be rare or dangerous in real-world settings. Throughout this thesis, the datasets used for the experiments are elaborated upon. Some visuals from the SemanticKitti dataset are shown in Figure 2.13.

CHAPTER 3

Integrating Synthetic Data with Real-World Data for LiDAR Segmentation

This chapter presents a hybrid LiDAR segmentation methodology that effectively combines synthetic and real-world datasets. The motivation toward this work was based on enhancing robustness and generalization of the segmentation models. In this work, synthetic data was created in a controlled autonomous driving simulation environment to complement real-world data from urban driving scenarios. It will be shown in this chapter by utilising spherical projection of the LiDAR point clouds and the incorporation of synthetic data and advanced deep neural network architectures boost the performance considerably. The experimental results underline the advantages of this hybrid

3.1 Motivation

Full autonomy in self-driving requires comprehensive scene understanding. Semantic segmentation-a task that assigns a relevant class label to each sensory data point-plays an important role in retrieving detailed situational awareness. For ex-

ample, the drivable area together with the position of preceding vehicles allow safe maneuver planning and decisions, strongly facilitating autonomous driving.

This would imply that autonomous vehicles should make valid and robust predictions, complemented by reliable estimates of uncertainty. For this, there is a need for numerous sensors capturing broad knowledge of the scene. The most absolute significance of redundancy in sensory data cannot be emphasized enough because the failure of one without a backup might lead to really catastrophic or financially significant outcomes. Furthermore, training and testing of ML models require much diverse fully labeled data Geiger et al. (2012).

During the last decade, DNNs have improved real-time semantic segmentation substantially both in accuracy and reliability. Even though most algorithms focus their efforts on processing camera images Poudel et al. (2019), Kendall et al. (2015), there is a growth of interest for the semantic segmentation of 3D LiDAR data Wu et al. (2018), Milioto et al. (2019). Unlike the camera images that are densely constituted with a grid structure, point clouds resulting from LiDAR are conventionally sparse, unstructured, and non-uniformly sampled. In spite of these challenges, LiDAR has greater field of view while measuring accurate distances, hence making it a crucial sensor in autonomous driving.

LiDAR scanners are one of the most deployed types of sensors in most autonomous driving systems; it performs very well regardless of changes in illumination. This is one of the major reasons for its soaring popularity in perception applications. These LiDAR scanners provide a point-cloud representation of the environment, and such representation is useful for high-level tasks in autonomous driving.

These point clouds can be utilised for a wide variety of tasks including but not limited to: point cloud segmentation Geiger et al. (2012), Qi, Su, Mo & Guibas (2017), 2D or 3D object detection Zhou & Tuzel (2018), Chen, Ma, Wan, Li &

Xia (2017), Qi et al. (2018), multi-modal fusion Zhou, Sun, Zhang, Anguelov, Gao, Ouyang, Guo, Ngiam & Vasudevan (2020), Jaritz et al. (2020), simultaneous localization and mapping (SLAM) Chen et al. (2019), Wu et al. (2018). This work focuses on the point cloud segmentation and attempts to label each point with its object type from the LiDAR scanner output. Effective segmentation of the point cloud is necessary for identifying road objects, such as pedestrians, vehicles, and lanes, which in turn are needed to accomplish tasks like autonomous driving. This chapter particularly targets identifying cars, cyclists, and pedestrians to enhance semantic understanding for the safe and efficient operation of an autonomous vehicle.

3.2 Related Work

Fully autonomous vehicle development has gained significant momentum over the past decade. Such an ambitious task requires overcoming a number of substantial challenges: small mistakes may have disastrous results. Large-scale and thoroughly annotated datasets play an important role in training and fine-tuning self-driving systems. Such big datasets have been proven to be very important in the learning of high-level image representations, especially with the popularity of deep learning methods. In the last years, several large datasets have been published targeting various aspects: lane tracking, semantic and instance segmentation, object tracking, and trajectory prediction. These datasets try to solve different complexities that computer vision systems face in the context of autonomous driving.

Cityscapes Cordts et al. (2016) provides a comprehensive benchmark suite to enhance semantic urban scene understanding, focusing on semantic and instance segmentation challenges. This dataset includes pixel-level fine annotations

and instance-level labels for vehicles and pedestrians, derived from diverse stereo video sequences collected across 50 cities. Neuhold et al. (2017) proposed the Mapillary Vistas dataset, which is larger and more diverse set of labeled data compared to Cityscapes. Ros et al. (2016) offers a means of obtaining realistic synthetic images with pixel-level annotations using a virtual environment. They generated the SYNTHIA dataset, including a vast collection of synthetic images for urban scene semantic segmentation, paired with automatically generated class annotations. Their experiments with deep convolutional neural networks demonstrated that incorporating SYNTHIA into the training process significantly enhances semantic segmentation performance on real-world images. The RobotCar dataset Maddern et al. (2017) includes data from six cameras, LiDAR, Global Positioning System (GPS), and an Inertial Navigation System (INS), collected over more than a year in central Oxford, aiming to create a benchmark suite akin to KITTI.

The KITTI dataset Geiger et al. (2013) includes the measurements of several sensors; among them are LiDAR and cameras. Since its creation, many different tasks and data were added to KITTI, and it became the ground for many experimental comparisons under various settings. Works like Wu et al. (2018), Aksoy et al. (2020) use KITTI for 3D LiDAR-point cloud segmentation. For the most part, there are RGB-D-based datasets that merge depth sensing with RGB data, which have driven a significant advance in the interpretation of point clouds, including semantic segmentation. ShapeNet Chang et al. (2015) stands out for point clouds of single objects, but this data may not transfer directly to other domains.

The KITTI dataset Geiger et al. (2013) includes data from several different sensors, such as LiDAR and cameras.Since its creation, many different tasks and data were added to LiDAR, and it became the ground for many experimental comparisons under various settings. Some studies, such as Wu et al. (2018), Aksoy et al.

(2020), use KITTI for 3D LiDAR point cloud segmentation. ShapeNet Chang et al. (2015) stands out for point clouds of single objects, but this data may not transfer directly to other domains. The Paris-Lille-3D dataset Roynard et al. (2018) provides aggregated scans with point-by-point annotations for 50 classes, 9 of which are used for evaluation. Behley et al. (2019) created the SemanticKITTI, which is the largest dataset focuses on 3D LiDAR point cloud segmentation, offering point-wise semantic and instance annotations for the KITTI point cloud sequences.

Recent advances in semantic segmentation of 3D LiDAR point clouds have leveraged deep neural network architectures Wu et al. (2018), Milioto et al. (2019), Qi, Su, Mo & Guibas (2017). These sophisticated methods differ not only in network architecture but also in the encoding of point cloud data. High-performance segmentation techniques typically employ fully convolutional networks Long et al. (2015), encoder-decoder architectures , or multi-branch models Poudel et al. (2019) in terms of network design. The main distinction among these architectures lies in how features are encoded at various depths and subsequently integrated to reconstruct spatial information.

There are two prevalent methods for representing irregular and unordered 3D LiDAR data: point-wise representation and projection-based rendering.

Point-wise approaches Pointnet Qi, Su, Mo & Guibas (2017), and Qi, Yi, Su & Guibas (2017) process LiDAR points without additional modifications or pre-processing. While effective for small point clouds, these techniques struggle with large LiDAR datasets without supplemental information from other sensors due to limited processing capacity. To facilitate point-wise processing, complementary cues such as camera image data have been successfully introduced Aksoy et al. (2020).

On the contrary, projection-based methods convert 3D point clouds into several different representations, such as voxel cells Zhou & Tuzel (2018) and multi-view

projections. Multi-view representation projects a 3D point cloud onto multiple 2D surfaces from different virtual camera perspectives, with each view processed by a multi-stream network. Voxel representation approaches convert a point cloud into a 3-dimensional volumetric form and assign each point to the relevant voxel Jaritz et al. (2020). But, the sparsity and irregularity of point clouds usually lead to repetitive calculations with voxelized data, as many voxel cells can remain empty. For efficiency, projecting 3D point clouds onto a 2D image plane and transforming point-cloud segmentation into conventional image segmentation has been suggested Wu et al. (2018). Subsequent refinements to the projection-based method have made it a preferred solution for large-scale point cloud segmentation Qi, Su, Mo & Guibas (2017), Milioto et al. (2019).

Unlike other projection-based and point-wise approaches, the 2D projection is more compact and dense, making it convenient for real-time computation and processing by typical 2D convolutional layers. Therefore, we project our 3D point cloud onto a 2D sphere prior to feeding it into the CNN.

3.3 Methodology

3.3.1 Dataset Creation

Creating a large dataset akin to ImageNet has significant challenges and is usually time-consuming. Therefore, synthetic data and simulation data become essential components in this endeavor. To produce our synthetic data for this study, an autonomous ego vehicle was equipped in the CARLA simulator with RGB, semantic, and depth cameras, as well as a semantic LiDAR scanner, all operating synchronously to generate matching data. The semantic LiDAR scanner, although lacking intensity information, provides point-wise instance and semantic ground-

truth data. The data collection occurred in Town-3, the largest and most complex city within the simulator. A top-down view of Town-3 is illustrated in Figure 3.1. Additionally, the city included other vehicles that drive autonomously and adhere to the Highway Code. Various sensors in the simulator were utilised to gather information on cars, cyclists, and pedestrians, enabling a wide range of research applications, including sensor fusion. The consistency between points and images also facilitates sanity checks on the collected data.

Figure 3.1: Top-down view of Town-3 in the CARLA simulator.

The rotation of the semantic LiDAR's laser rays was simulated using ray casting, with 64 vertical channels producing raw data that includes the point's coordinates, angle of incidence, surface normal, instance, and semantic ground-truth. The sensor's horizontal rotation was calculated, and ray casting was performed to generate a 3D point cloud at each step. With a rotation frequency of 10 Hz and a

vertical field of view of 26.9 degrees (+2 degrees for the upper and -24.9 degrees for the lower), the sensor emits over 1.3 million points per second, covering a range of up to 100 meters, similar to the capabilities of the Velodyne HDL-64E.

The CARLA simulator was used to generate a synthetic dataset comprising over 8000 samples. This dataset includes a number of point clouds comparable to the 8545 point clouds in the KITTI training set, as well as RGB images, semantically segmented images, depth images, and their associated 3D LiDAR point clouds.

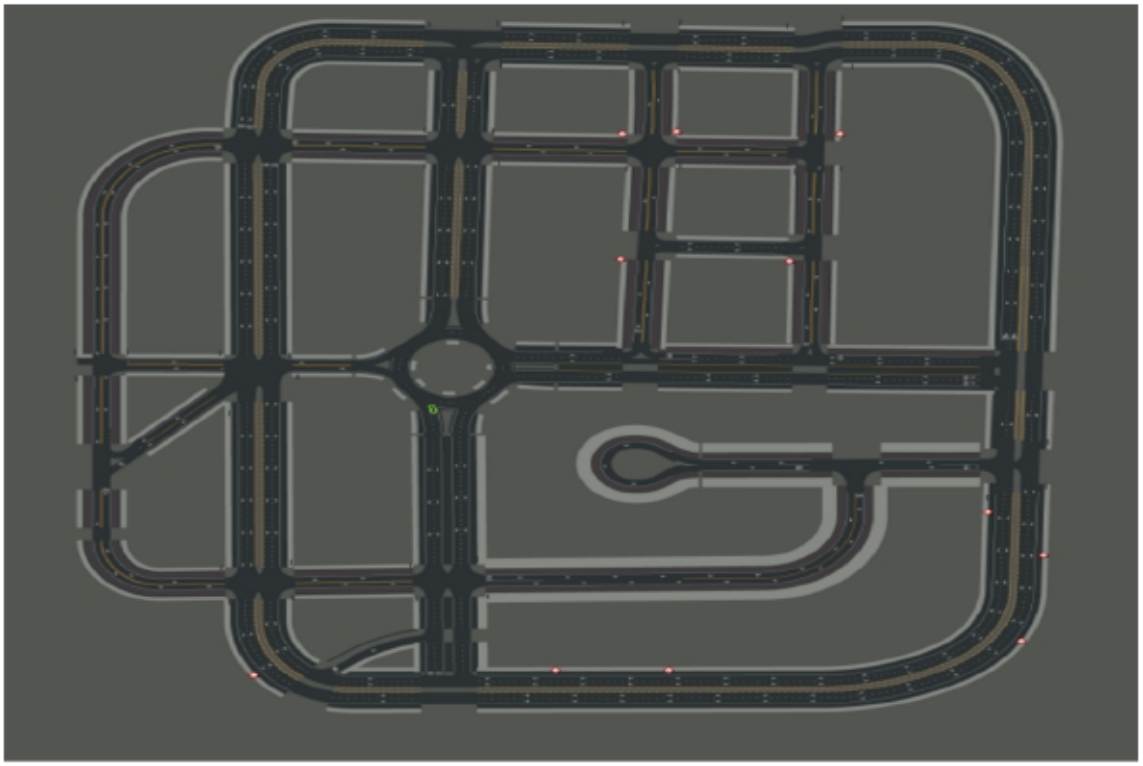

Figure 3.2: CARLA Town3 Topology.

The Town-3 topology map can be seen in Figure 3.2, providing an overview of the simulation environment used for data collection. Figure 3.3 shows Town-3 recorded from different angles, offering additional perspectives of the simulated city.

Figure 3.4 illustrates the vehicles detected with bounding boxes, demonstrating

Figure 3.3: CARLA Town3 from different angles.

Figure 3.4: Vehicles detected with bounding boxes in Town-3.

the capability of the detection system within the simulated environment.

Figure 3.5 shows several different image types recorded during the dataset generation process. The first two columns display the screens captured by the RGB camera. The two columns in the middle depict the semantic segmentation images corresponding to the RGB ones, depth images captured by the depth camera are given in the final two columns.

A point cloud generated by the LiDAR scanner in the virtual environment is given in Figure 3.6, illustrating the detailed information captured by the sensor.

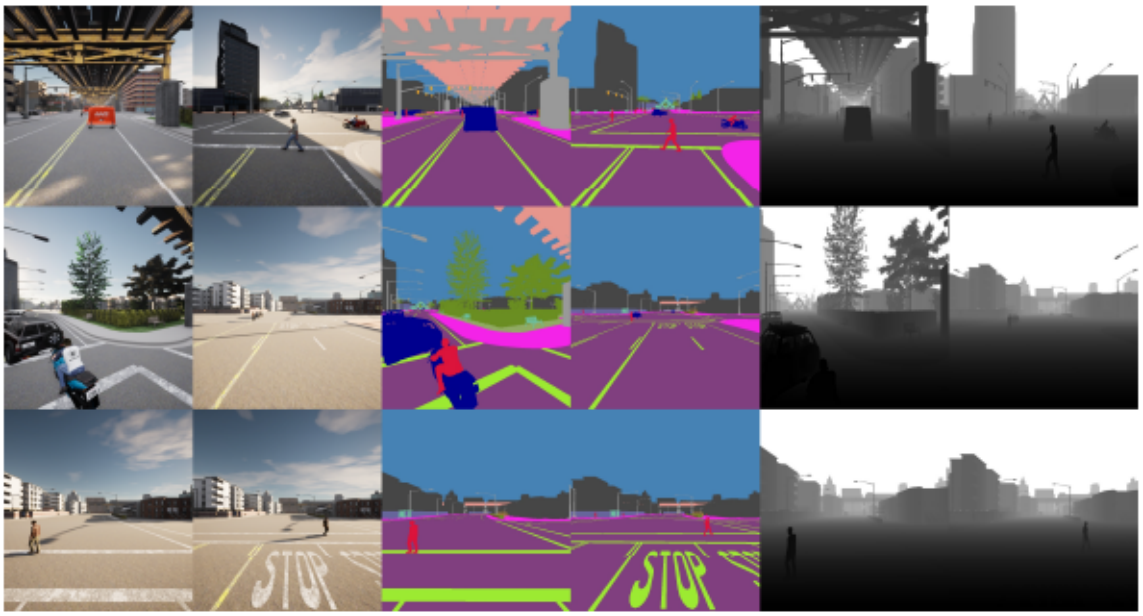

Figure 3.5: Some images recorded during the dataset creation process with different sensors. The first 2 columns show the screens captured by the RGB camera mounted on the vehicle. The middle columns show the semantic segmentation images corresponding to the RGB images. The last 2 columns display the depth images corresponding to the RGB images collected by the depth camera.

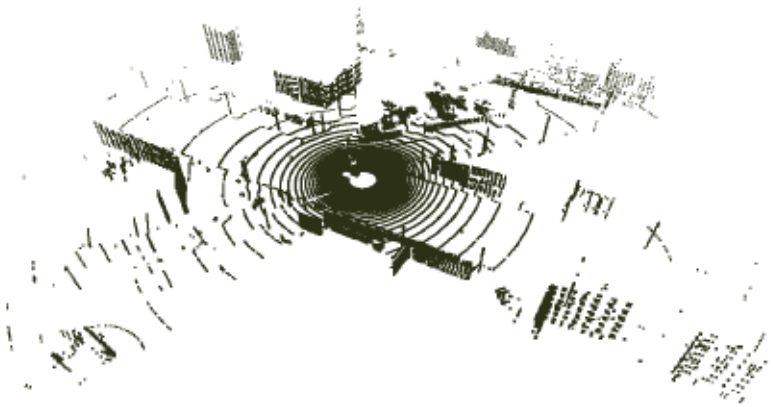

Figure 3.6: Example point cloud obtained from the simulation environment.

Table 3.1: LiDAR parameters.

Parameters	Values
Channels	64
Range	100 m
Points per second	1.35 m
Rotation Frequency	10 Hz
UpperFovLimit	2.0°
LowerFovLimit	−24.9°

Figure 3.7 shows an RGB image captured during the dataset creation process, and also displays its corresponding semantic LiDAR point cloud. These figures highlight the multi-sensor data collection and the alignment between different sensor modalities.

(a) RGB Image (b) LiDAR Point Cloud

Figure 3.7: RGB camera image and the corresponding LiDAR point cloud captured from the simulator.

The LiDAR parameters, as applied within the simulation environment, are given in Table 3.1. These include, but are not limited to the number of chan-

nels, range, points per second, rotation frequency, and the field of view limits. The setup below presents accurate and reliable data collation for the synthetic dataset.

Classifications for cyclists were one of the more noticeable challenges while creating this dataset. In the simulation system, due to a lack of a separate class for cyclists, they fall under the category of a vehicle. Fundamentally, this cannot be said to be a limitation in the simulation software, as it reflects the inadequateness present in the classification system itself. To overcome this, we developed a method to distinguish cyclists from cars within the simulation. This will enable us to separate data belonging to the cyclists and carry out an analysis of them in greater detail, with respect to their behavior and movements in this virtual environment.

3.3.2 Spherical Projection of the LiDAR Point Cloud

Conventional convolutional neural network architectures work with images that can be represented as a three-dimensional tensor with dimensions $H \times W \times C$. Here, H and W represent the height and width, respectively, while C represents the number of encoded features, such as RGB values when $C = 3$. LiDAR point clouds comprise of cartesian coordinates x, y, z and can also include features such as intensity and RGB values. But, the distribution of LiDAR point clouds differs from typical RGB image pixels, as it is typically sparse and irregular. Some research approaches handle this by discretising the data into voxels and engineering features like disparity, mean, and saturation. One of the challenges with directly manipulating point cloud data is the absence of a proper ordering, which complicates the learning of order-invariant feature extractors. As a result, one can infer that naively discretising a 3D point cloud into voxels results in many empty voxels, resulting in inefficient processing and redundant computation.

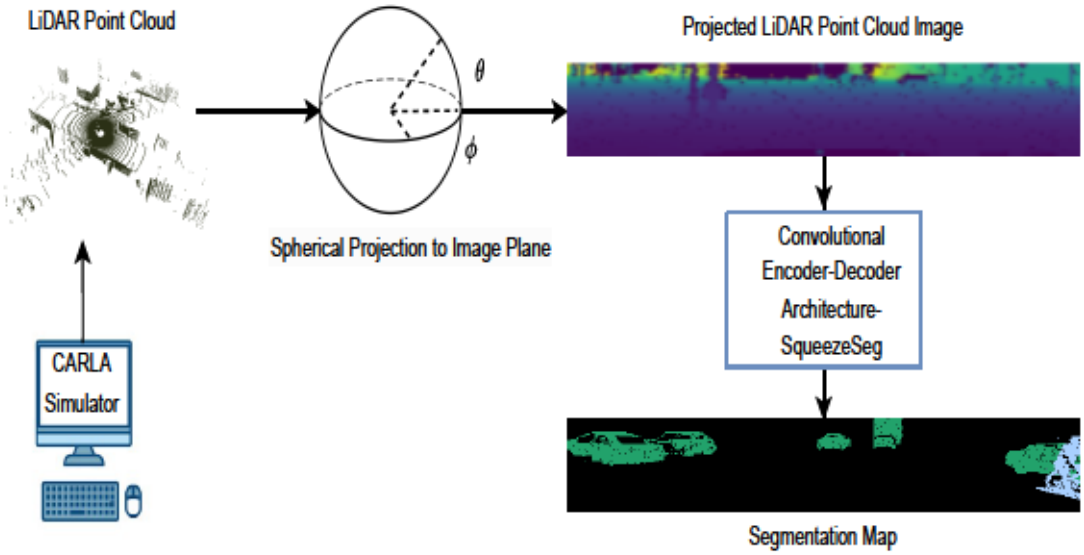

Figure 3.8: Steps followed during the work can be summarized as the diagram above. We have used CARLA autonomous driving simulator to collect our synthetic data.

In order to achieve a more compact representation and tackle the aforementioned problem, Kendall et al. (2015) projects LiDAR point cloud data onto a sphere. This operation converts the sparse, irregular data into a dense, grid-like format. The LiDAR scanner's rotating shape resembles a hollow sphere, with a projection equation as follows:

$$\theta = \arcsin\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right), \quad \tilde{\theta} = \left\lfloor \frac{\theta}{\Delta\theta} \right\rfloor \tag{3.1}$$

$$\phi = \arcsin\left(\frac{y}{\sqrt{x^2+y^2}}\right), \quad \tilde{\phi} = \left\lfloor \frac{\phi}{\Delta\phi} \right\rfloor \tag{3.2}$$

In these equations, θ and ϕ are the azimuth and zenith angles, respectively, as illustrated in Figure 3.8. The $\Delta\theta$ and $\Delta\phi$ discretization resolutions and $(\tilde{\theta}, \tilde{\phi})$

are the coordinates of a point on a two-dimensional spherical grid. This approach creates a 3D tensor with dimensions $H \times W \times C$ by applying this projection to each point in the LiDAR data.

In our study, data is collected from a semantic LiDAR using CARLA simulator, which has 64 channels. We consider a 90-degree field of view and divide it into 512 grids, with $W = 512$. C denotes the number of features associated with each point. Each LiDAR point includes four features: (x, y, z) coordinates and point intensity, with a range $r = \sqrt{x^2 + y^2 + z^2}$. By processing the data in this manner, point cloud segmentation can be treated as a traditional image segmentation task handled using standard convolutional neural networks.

3.3.3 Network Architecture

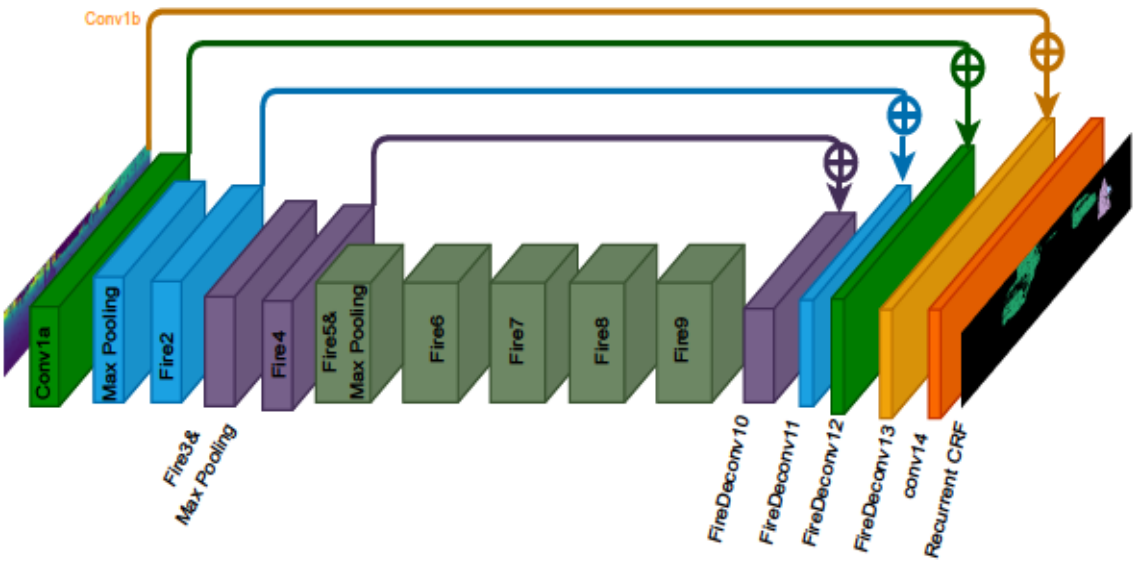

Figure 3.9: Adopted from Wu et al. (2018) convolutional encoder-decoder neural network architecture with a CRF at the end is given in this figure.

SqueezeSeg, a derivative of SqueezeNet Iandola et al. (2016), is a lightweight

CNN that achieves accuracy comparable to AlexNet Krizhevsky et al. (2012) with significantly fewer parameters, approximately 50 times less. It offers a reliable, fast, and comprehensive end-to-end approach for accurately segmenting road objects from LiDAR point clouds.

SqueezeSeg processes an input tensor of dimensions $64 \times 512 \times 5$. The layers of SqueezeNet, ranging from conv1a to fire9, are adapted for feature extraction purposes. Fire module is shown in Figure 3.10. SqueezeNet uses max-pooling layers to downsample intermediate activation maps in both width and height dimensions. But, since the height of the input tensor is notably smaller than its width in a projected point cloud, downsampling is limited to the width dimension only. The fire9 layer outputs a downsampled feature map that encapsulates the semantics of the point cloud. In order to upsample the activation maps in the width dimension

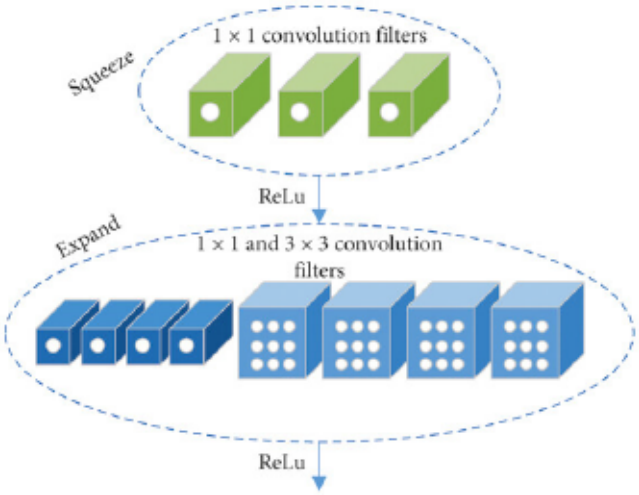

Figure 3.10: Fire module Iandola et al. (2016).

and generate label predictions with full resolution for each point, deconvolution modules (more accurately termed "transposed convolutions") are employed. As it is shown in Figure 3.9, skip-connections are utilised to merge upsampled feature

maps with lower-level feature maps of identical size. The final output probability map is produced by a convolutional layer (conv14) equipped with a softmax function. A recurrent Conditional Random Field (CRF) layer further refines this probability map.

During image segmentation process, CNN models tend to lose low-level information in the process of downsampling, which ends up with label maps lacking clear boundaries. This behavior is also observed in the proposed network, necessitating the integration of both high-level semantics and low-level features to achieve accurate point-wise label predictions.

3.3.4 Conditional Random Field

Capturing both the high-level context and low-level details of an object and scene are necessary to have accurate label prediction. Providing label consistency is crucial, such as assigning the same label to points with similar intensity and proximity. A CRF is utilised to enhance the label map generated by the CNN. The CRF model employs an energy function given in Equation 3.3 to refine label predictions for a given point cloud, where c_i represents the predicted label of the i-th point in the cloud.

$$E(c) = \sum_i u_i(c_i) + \sum_{i,j} b_{i,j}(c_i, c_j), \tag{3.3}$$

The "unary potential" term, $u_i(c_i) = -\log P(c_i)$, reflects the CNN classifier's predicted probability $P(c_i)$. The binary potential terms describe the "penalty" for assigning different labels to points that are otherwise similar, expressed as $b_{i,j}(c_i, c_j) = \mu(c_i, c_j) \sum_{m=1}^{M} \omega_m k^m(f_i, f_j)$, where $\mu(c_i, c_j) = 1$ if $c_i \neq c_j$ and 0 otherwise, k^m is the m-th Gaussian kernel dependent on the features f of points i and

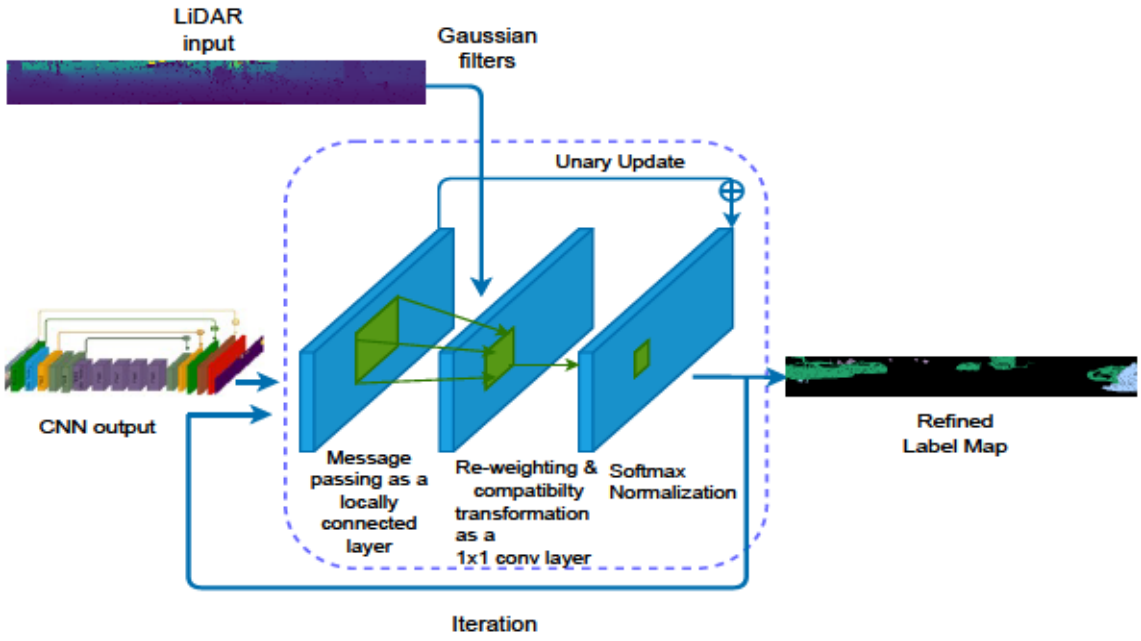

Figure 3.11: Conditional Random Field structure implemented as a recurrent neural network layer.

j, and ω_m is the corresponding coefficient. Similar types of Gaussian kernels have been utilised in Wu et al. (2018).

$$w_1 \exp\left(-\frac{\| p_i - p_j \|^2}{2\sigma_\alpha^2} - \frac{\| x_i - x_j \|^2}{2\sigma_\beta^2}\right) + w_2 \exp\left(-\frac{\| p_i - p_j \|^2}{2\sigma_\gamma^2}\right), \tag{3.4}$$

The first term is influenced by both the angular position $p(\tilde{\theta}, \tilde{\phi})$ and the cartesian coordinates (x, y, z) of two locations, while the second term is affected only by the angular positions. $\sigma_\alpha, \sigma_\beta$, and σ_γ are three empirically chosen hyperparameters. Additional features, such as intensity and RGB values, can also be included. A refined label assignment is achieved by minimizing the CRF energy function. An overview of the mean-field iteration implementation as an RNN module is presented in Figure 3.11. The initial probability map for the CRF module is derived

from the CNN model's output.

3.4 Evaluation Outcomes

3.4.1 Evaluation Metric

For class-level segmentation tasks, our model performance is evaluated based on each predicted point label against its corresponding ground truth annotation. Precision (P), Recall (R), and intersection-over-union (IoU) values are reported our evaluation measurements:

Precision is the measure of the accuracy of the positive predictions of the model. That is, the ratio of correctly predicted positive observations to all the observations that are predicted positive. It is defined as follows:

$$Pr_i = \frac{\mid P_i \cap G_i \mid}{\mid P_i \mid}, \tag{3.5}$$

where P_i is the predicted point set of class i and G_i is the equivalent ground truth set.

Recall evaluates the model's ability to identify all relevant instances, rep-resenting the proportion of correctly predicted positive instances among all actual positive instances. It is defined as follows:

$$Recall_i = \frac{\mid P_i \cap G_i \mid}{\mid G_i \mid}, \tag{3.6}$$

where P_i is the predicted point set of class i and G_i is the equivalent ground truth set.

Intersection-over-Union (IoU) basically calculates the measure of overlap between the prediction and ground truth sets, providing more information on how much of their intersection belongs to the union. It is formally defined as follows:

$$IoU_i = \frac{|P_i \cap G_i|}{|P_i \cup G_i|}, \tag{3.7}$$

where $|\ .\ |$ is the total number of points in a set, P_i represents the predicted point set of class i, and G_i is the corresponding ground truth set.

We can also define precision and recall in terms of True Positives (TP), False Positives (FP), and False Negatives (FN) to make it clearer:

Precision can be defined as the ratio between true positive instances and the value obtained by summing true positive and false positive instances:

$$P_i = \frac{TP_i}{TP_i + FP_i}, \tag{3.8}$$

where TP_i represents true positives, and FP_i represents false positives for class i.

Recall is defined as the ratio of true positive instances to the total sum of true positive and false negative instances:

$$R_i = \frac{TP_i}{TP_i + FN_i}, \tag{3.9}$$

where TP_i represents true positives, and FN_i represents false negatives for class i.

We primarily use the IoU score as our main accuracy metric in our experiments.

3.4.2 Results

Figure 3.12 shows a loss value for every training step of the model. The x-axis shows batch step number ranging from 0 to 30,000, while the y-axis represents the loss value. Figure 3.13 illustrates the progression of the loss value during the validation phase, providing insight into how well the model generalizes to unseen data.

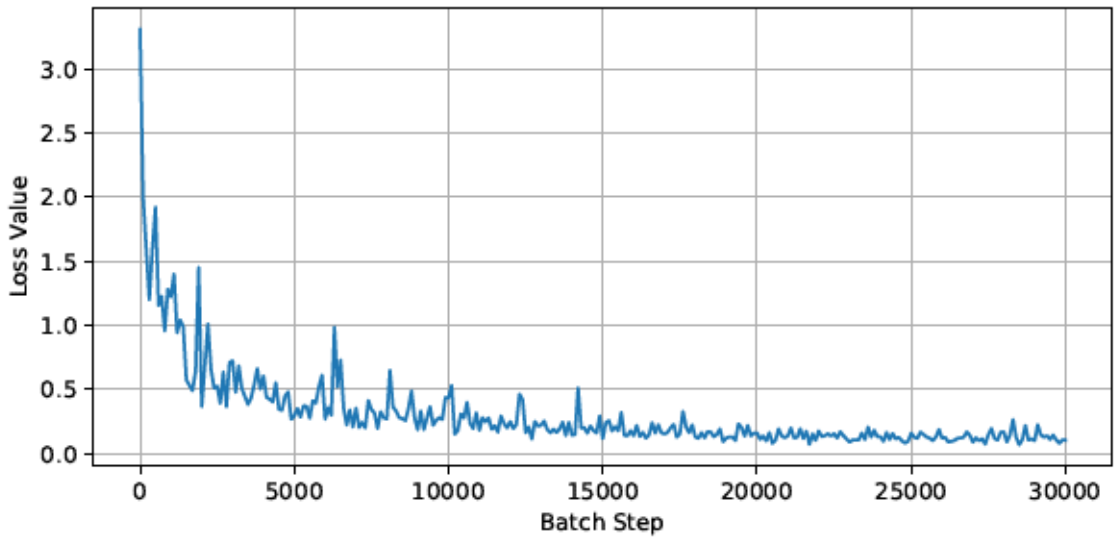

Figure 3.12: Training loss per batch step observed during the process.

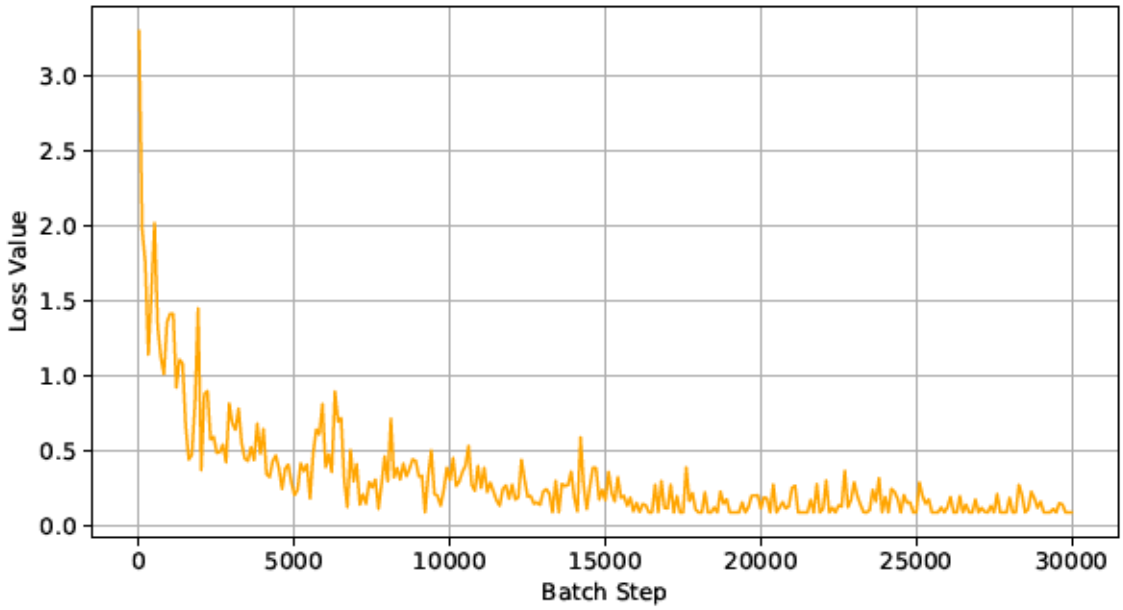

Figure 3.13: Validation loss per batch step observed during the process.

The initial value of the loss is high, as expected, hence representing that the model is far from correctly predicting. As training continues, the loss value starts

to decrease further, showing that the model actually learns from the data to present better results. The model's capability to reduce the error margin in the prediction is exactly shown in the graph with a decreasing trend, effectively signaling that the model is learning. Comparison of training and validation loss curves demonstrates that overfitting is not occurring in the model. Both curves exhibit a consistent downward trend, reflecting effective learning during training. Although the training loss decreases slightly faster, the validation loss closely follows with minor fluctuations, which is expected due to the variability of unseen data. Importantly, the validation loss does not diverge significantly from the training loss at any point, indicating that the model generalises well to unseen data.

A reduced loss value indicates an improved accuracy of the model over time as it learns to make better predictions based on the training data. Ideally, the goal is for this value of loss to converge into stability at a minimum value, which would mean the model has been sufficiently trained and can make correct predictions on newer, unseen data.

The results for only the KITTI dataset are presented in Table 3.2. As mentioned in the previous section, our primary evaluation metric is IoU.

Table 3.2: Network's segmentation performance on KITTI data. The table includes metrics for precision (P), recall (R), and intersection-over-union (IoU), with IoU being the primary measure of accuracy. All values are expressed as percentages.

KITTI			
	P	**R**	**IoU**
Car	58.9	95.0	57.1
Cyclist	35.7	45.8	25.1
Pedestrian	45.2	29.7	21.8

Table 3.3: Network's segmentation performance on KITTI data combined with our synthetic data. All values are expressed as percentages.

KITTI+CARLA			
	P	**R**	**IoU**
Car	70.2	99.3	**69.9**
Cyclist	44.5	87.3	**44.4**
Pedestrian	38.7	84.1	**38.5**

Our synthetic point cloud data collected from the CARLA simulator does not include intensity information. To examine the impacts of training with synthetic data, the model was initially trained on the KITTI training set without including intensity values and validated on the KITTI validation set. Subsequently, we combined our CARLA dataset with the KITTI training data and retrained the model.

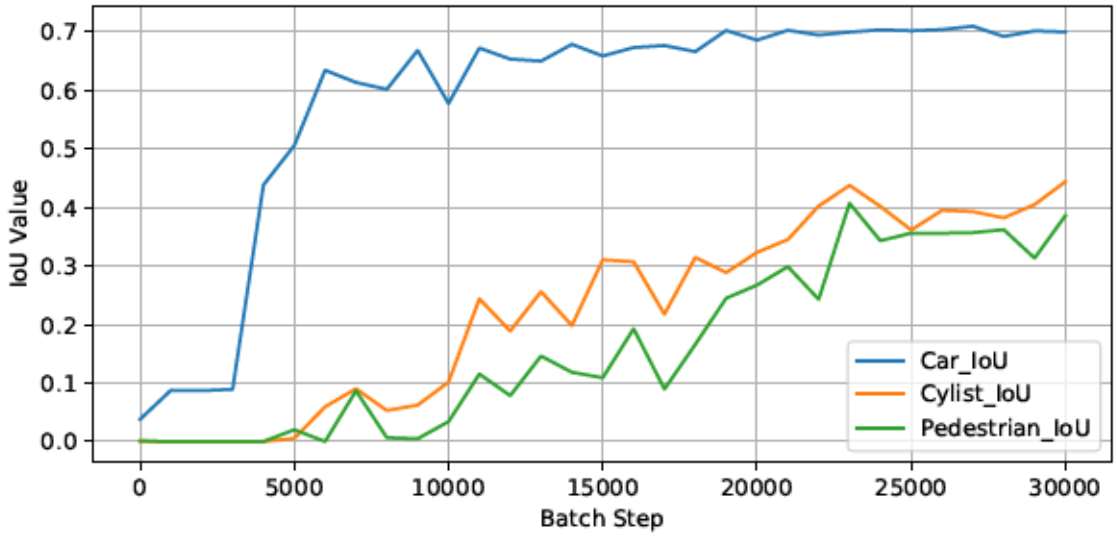

Figure 3.14: IoU values of the object classes when an equal amount of synthetic data as the amount of real data is used.

The results over integrated dataset are shown in Table 3.3. IoU values throughout the training process are shown in Figure 3.14. The network's segmentation performance for car, cyclist, and pedestrian classes improves significantly with the addition of our synthetic data, as shown in Table 3.2 and Table 3.3. The results indicate a notable boost in accuracy compared to training solely on the KITTI dataset.

Furthermore, when compared to KITTI+GTA5 (GTA5 is another synthetic dataset which is collected from the GTA5 video game) data as is shown in Table 3.4, which did not include cyclist and pedestrian classes, our synthetic dataset enhances the network's performance and outperforms it. Figure 3.15 displays a visualization comparing the segmentation output of the network we trained on the combined KITTI and our synthetic point cloud data with the ground truth labels.

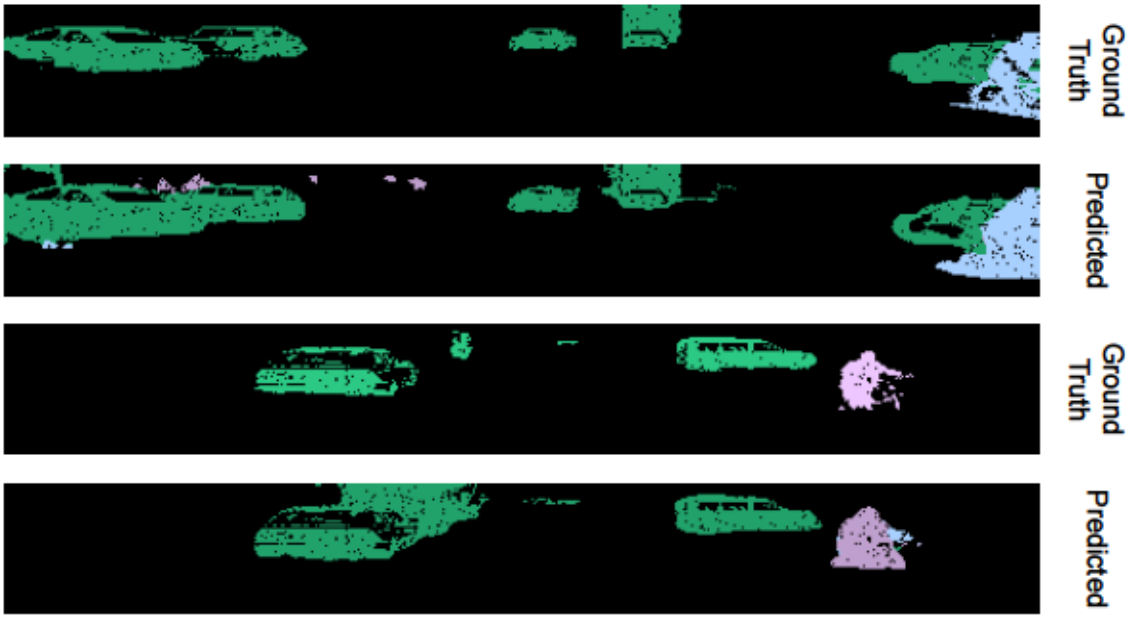

Figure 3.15: Predicted labels are compared to actual, correct labels. The network accurately identifies objects not present in true labels, in addition to correctly identifying labeled objects.

Table 3.4: Comparing network segmentation performance on combined KITTI and our data with KITTI+GTA5 data on proposed network and validated on KITTI validation data.

	KITTI+GTA5 Data			KITTI+CARLA		
	P	R	IoU	P	R	IoU
Car	68.6	92.8	66.0	70.2	99.3	**69.9**
Cyclist	N/A	N/A	N/A	44.5	87.3	**44.4**
Pedestrian	N/A	N/A	N/A	38.7	84.1	**38.5**

3.4.3 Ablation Study

In this ablation study, we investigated the effect of the amount of simulation data on the network performance. By training the network with 50% less synthetic data than real data, we achieved the results in Table 3.5 and Figure 3.16. In particular, the results in Table 3.5 and Figure 3.16 highlighted that even with fewer synthetic data, the performance of the network outperformed results obtained by training the network only on the KITTI dataset.

Table 3.5: Segmentation performance of the network on the classes using only an amount of 50% synthetic data together with the real data and its comparison to network's performance on only real world data.

	KITI Data			KITTI+CARLA		
	P	R	IoU	P	R	IoU
Car	58.9	95	57.1	62.7	95.7	**65.6**
Cyclist	35.7	45.8	25.1	37.5	55.8	**31.1**
Pedestrian	45.2	29.7	21.8	37.5	44.6	**27.1**

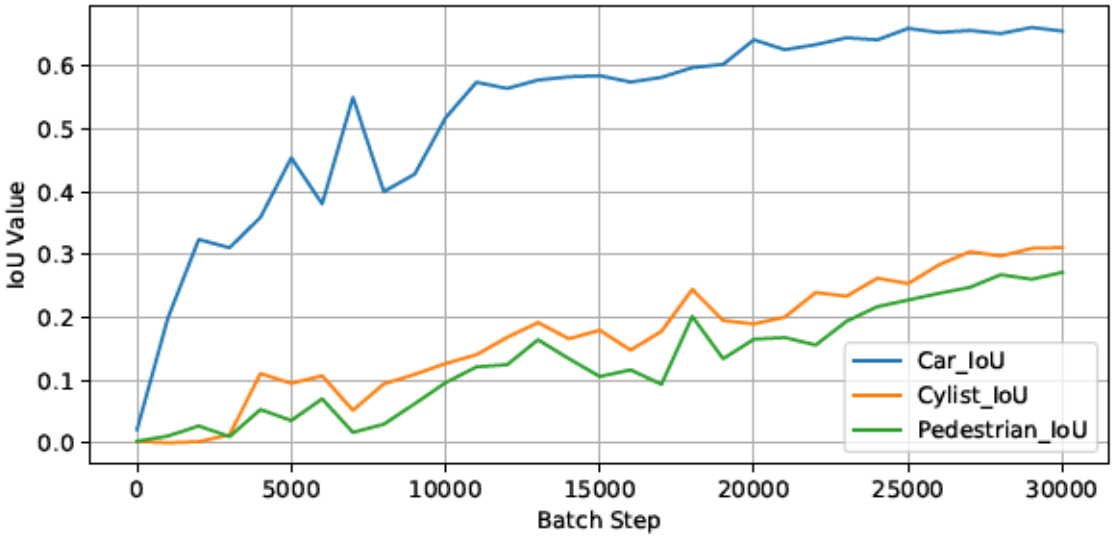

Figure 3.16: IoU values of the classes with less amount, 50%, of synthetic data.

Note that accuracy values decreased compared to training on an equal amount of synthetic and real data. This proves that the synthetic data will significantly enhance the network's accuracy, even though its benefits diminish when drastically lowering the quantity of synthetic data. The tables below show a few specific, class-by-class comparisons of network performance metrics: precision, recall, IoU.

3.5 Conclusion

This chapter had contributions to the integration of synthetic data with real data in capturing better neural network accuracies within the context of autonomous driving. Deep neural network architecture was modified in order to combine synthetic data from simulator with real-world point cloud data. Experimental results showed that this hybrid training, further validated on real-world KITTI data yields significant performance improvements compared to previous methodologies that employed GTA-5 data. This probably is because of reduced noise in our synthetic

dataset. Also, the fact that we focused on the main task of classifying cars, cyclists, and pedestrians significantly improved segmentation accuracy.

The findings of this chapter underscore the importance of leveraging synthetic data to address data scarcity and variability in real-world scenarios. The demonstrated ability of synthetic datasets to enhance segmentation performance opens opportunities for broader applications, including the classification of additional objects like vegetation or infrastructure elements. This work highlights the utility of hybrid training paradigms for improving object differentiation and classification in complex transportation environments.

The insights gained from this chapter laid the foundation for exploring innovative approaches in autonomous driving perception tasks, shaping the research directions presented in subsequent chapters. Specifically, the success of hybrid training motivated a deeper investigation into architectures that can model complex spatial relationships across diverse datasets. The need for robust feature extraction, evident in this chapter, inspired the adoption of VViTs for their capability to capture both local and global context in point cloud segmentation tasks. This transition builds on the findings about leveraging data diversity and architectural modifications to enhance segmentation tasks. By building on the methods and challenges addressed here, the next chapter delves into leveraging advanced architectures to further refine segmentation performance. Furthermore, the exploration of synthetic data integration influenced our approach to multi-object tracking and motion forecasting by demonstrating the value of hybrid datasets for improving accuracy and robustness.

Future research can expand upon the contributions of this chapter in several meaningful ways. Refining the synthetic data generation process to cover a wider range of environmental conditions and traffic scenarios could further enhance model generalization. Additionally, integrating multi-modal data presents an av-

enue for capturing a more holistic view of the scene. Such efforts could not only improve segmentation accuracy but also lay the groundwork for better perception systems capable of handling diverse real-world conditions.

Beyond segmentation, this chapter also lays the foundation for advancing autonomous driving systems' perception pipelines. The method discussed here provide a stepping stone for addressing downstream tasks such as tracking and motion forecasting. The hybrid training strategies and architectural modifications developed here align with the broader goal of creating reliable, robust perception frameworks for autonomous systems, contributing directly to safer and more efficient driving solutions.

In summary, this chapter demonstrated the potential of synthetic data integration in improving segmentation accuracy and introduced techniques that address critical challenges in real-world applications. By linking the findings here with subsequent advancements in architectural exploration, this work sets the stage for further innovations discussed in later chapters, forming a cohesive narrative in the development of advanced perception systems for autonomous driving.

CHAPTER 4

LiDAR Segmentation and Vision Transformers

This chapter introduces an advanced framework for segmenting LiDAR point clouds using vision transformers that can enhance state-of-the-art autonomous driving systems. Different from traditional approaches, which often rely on the use of CNNs, this approach will exploit the strengths of ViTs in modeling challenging spatial dependencies without requiring pre-training. This framework projects the unstructured 3D LiDAR data into structured 2D format space using spherical projection techniques, thus making it appropriate for the ViTs. Besides, the performance of the proposed method is further evaluated on a multitask learning setup: two jointly trained ViTs learn feature representations and perform segmentation tasks with high potentials for improved accuracy in real driving scenarios. This chapter highlights the trend of advantages of ViTs in the handling of LiDAR data and emphasizes that they can outperform traditional approaches.

4.1 Motivation

The self-driving car has to have an in detail understanding of its surroundings to gain absolute autonomy. Semantic segmentation has a huge impact on this that it categorizes the input from various sensors that highlights the safe route to be taken by the vehicle in such a way that other vehicles around it are located. This capacity is expected by the artificial intelligence of the vehicle in order to make decisions effectively through appropriate interpretation of the real world. Moreover, the collection of a large amount of annotated data from different types is an essential pre-requisite for the training as well as validating phases of many applications' ML models in order to make their performances robust in dynamic and complex situations Geiger et al. (2012).

In recent years, much attention has been devoted to real-time semantic segmentation, while the evolution of deep neural networks has substantially improved the accuracy and reliability in this field. Segmentation performance in most of the presently proposed algorithms relies on the dense, structured nature of the camera images' pixel grid Poudel et al. (2019), Kendall et al. (2015). Semantic segmentation on 3D LiDAR data is comparably less explored to date as opposed to image-based approaches Wu et al. (2018), Milioto et al. (2019).In the case of LiDAR point cloud data, including rich representation of a 3D environment, it is sparse, unstructured, and noisy; hence, its direct processing in view of segmentation tasks poses a challenging task.

Overcoming these challenges required various ways of converting the raw data from point clouds into more structured forms like voxel grids and mesh models. Recently, owing to its efficiency in LiDAR handling for semantic segmentation, spherical projection has attracted much attention. Such an approach simplifies

such complicated 3D data into a 2D format that is more manageable for processing by neural networks. Historically, CNNs have dominated the architecture in image recognition and served as the foundational approach in the field of computer vision Krizhevsky et al. (2012). Inspired by the human visual system itself, CNNs perform impressively on object detection and semantic segmentation Long et al. (2015), both of which require comprehension and classification of image data. More recently, the family of models that were considered successful was challenged by the emergence of a new paradigm with so-called ViTs, which have given a new impulse in the computer vision area by applying the principles of transformer models—originally designed for natural language processing—to the analysis of images Dosovitskiy et al. (2020).

Procedures involved in ViTs differ from the procedures involved in CNNs, because, while all pixels are processed simultaneously and not hierarchically, it allows a broader and more global perception of the content of an image Carion et al. (2020). This property enables ViTs to model long-range dependencies with variable-length inputs, as it has never been easier before; therefore, it makes them highly suitable for most visual tasks. Indeed, ViTs have given state-of-the-art results on many benchmarks outperforming CNN in several domains: image classification, object detection, semantic segmentation Zhang et al. (2021). However, here it is worth mentioning that ViT needs a large dataset for their training because they tend to be more data dependent Touvron et al. (2021).

With these considerations in mind, exploring applications of ViTs to the area of autonomous driving, and, in particular, with LiDAR data for semantic segmentation, presents great avenues of research. That is, the high-resolution capability of LiDAR sensors combined with sophisticated dependency modeling by ViTs have the potential to significantly enhance the accuracy of segmentation and overall performance of self-driving vehicles.

In this chapter, we introduce a new approach for point cloud segmentation, which is based on the combination of spherical projection methods with Vision Transformer networks. Our methodology bridges 3D LiDAR data into 2D images by spherical projection and thus opens the door for segmenting tasks using a class of ViTs that have demonstrated state-of-the-art performance in numerous computer vision applications Milioto et al. (2019). The idea of combining both techniques is to provide a robust and adaptive framework of point cloud segmentation, increasing the accuracy and efficiency of autonomous driving systems. The three stages from the proposed methodology are depicted in Figure 4.1.

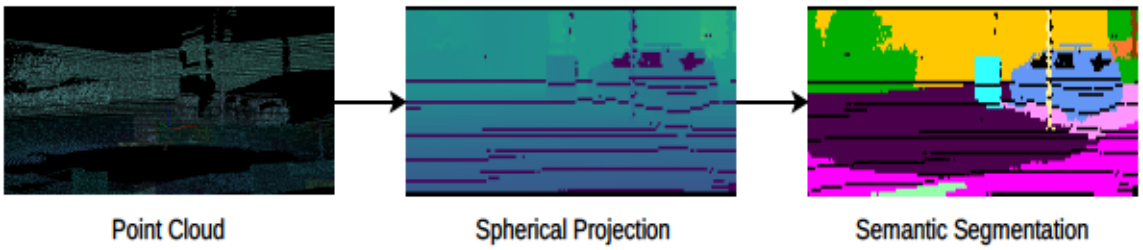

Figure 4.1: Block diagram of the approach.

4.2 Related Work

Deep learning has recently gained considerable momentum for solving point cloud tasks, including segmentation. Initial efforts were primarily put into image-based segmentation methods. However, due to the rapid growth of LiDAR technology, it is now well established that the 3D point clouds capture depth information with unprecedented accuracy. This fact finally brought real attention to the peculiar difficulties of LiDAR data: a scarcity of large-scale datasets and computational expenses for handling it.

The usual way to handle them is by voxelizing the point cloud data and then

taking the 3D CNN in for further processing data. These representation models possess good spatial relation modeling within the voxel grid, hence, enabling one to segment 3D space more effectively. However, the voxel-based methodology also comes with its problems; it typically has highly dimensional issue and computational overhead is tremendous, finding applications nowhere near real time.

Other alternatives could project the point cloud onto a 2D plane, similar to creating an RGB image, and perform segmentation using 2D CNNs. This whole procedure lets the computation be efficient, decreasing the complex processing of 3D information. However, it has the drawback of having the tendency to lose important 3D spatial information relevant for understanding the complete scene geometry. In the short term, works have introduced this weakness by combining the representations from both 2D and 3D data-point projections on multiple 2D planes from different perspectives. These projected and concatenated images are first processed by 2D CNNs that leverage both depth and spatial context, therefore providing a wider contextual comprehension of the environment.

4.2.1 LiDAR and Semantic Segmentation

In recent years, semantic segmentation has progressed quite a lot, especially for the domain of autonomous driving, where correctly understanding the scene is crucially important. Much of this improvement owes a debt to two factors: deep learning methodologies, and huge datasets, such as CamVid Brostow et al. (2009), Cityscapes Cordts et al. (2016), and Mapillary Vistas Neuhold et al. (2017). These have paved the way for sophisticated neural network models such as Deeplab V3 Chen, Papandreou, Schroff & Adam (2017) and PSPNet Zhao et al. (2017) which would provide high-quality segmentation output.

Due to the fact that prior to large-scale LiDAR datasets, methods for point

cloud segmentation were comparatively less developed. Recently, work pioneering in this arena was done by Wu et al. (2018), by developing the SqueezeSeg network. This was further fine-tuned and improved to create the SqueezeSegV2 Wu et al. (2019) and SqueezeSegV3 Xu, Wu, Wang, Zhan, Vajda, Keutzer & Tomizuka (2020) models. These were based on bounding box annotations present in the KITTI dataset Geiger et al. (2012) and synthesized scans from game engine simulations to train superior segmentations.

These paved the way for further research in this area, as evidenced by the recent publication of SemanticKITTI dataset Behley et al. (2019)—a fully annotated large dataset for LiDAR scans based on KITTI odometry dataset. Indeed, the dataset has catalyzed intensive research in a variety of machine learning algorithms with regard to point cloud segmentation, showing marked performance improvements. Notable architectures to highlight are PointNet Qi, Su, Mo & Guibas (2017), PointNet++ Qi, Yi, Su & Guibas (2017), and SalsaNext Cortinhal et al. (2020) in which innovative approaches have sought to address the inherent problem of non-structure and order in point cloud data. However, these networks still have a high computational cost regarding the processing of 3D point clouds for real-time applications.

4.2.2 Vision Transformers in Computer Vision

Work by Dosovitskiy et al. (2020) introduced vision transformers, another step away from sole reliance on CNNs in computer vision. ViTs are based on the transformer architecture developed for natural language processing; they process images as sequences of patches rather than collections of individual pixels. This enables the ViT to capture a representation of the whole image, which works especially well in tasks such as semantic segmentation. A schematic overview of the

model is given in Figure 4.2.

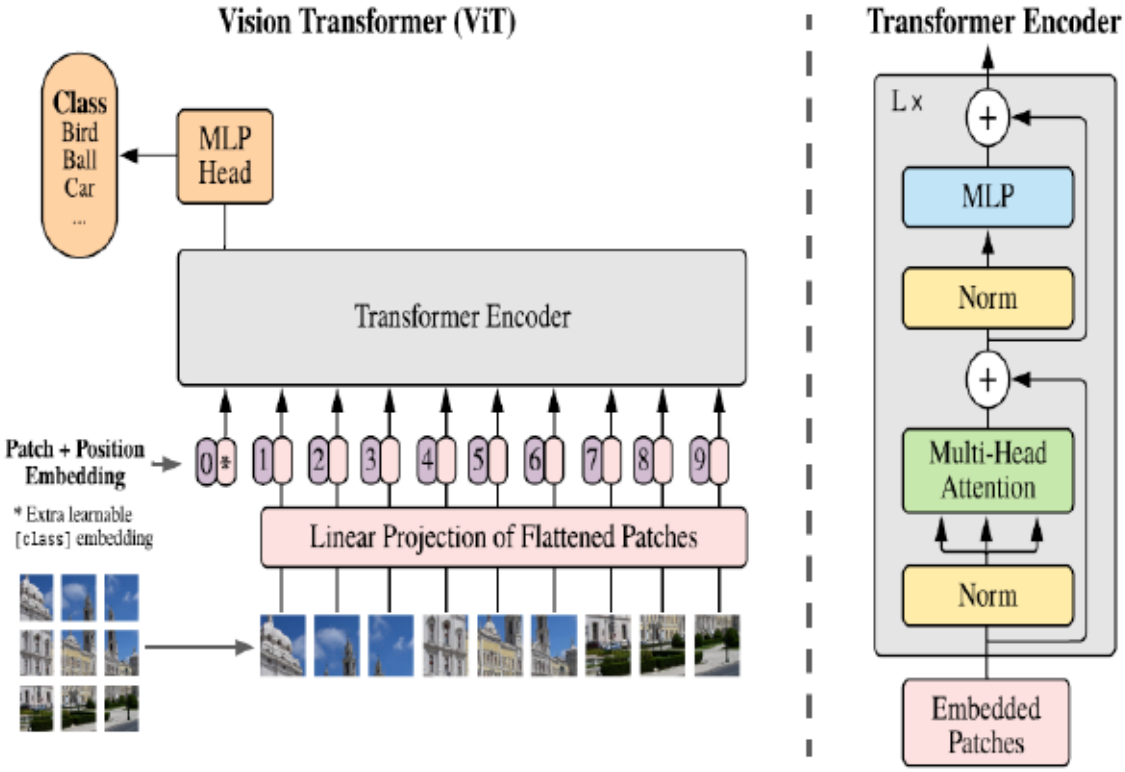

Figure 4.2: Vision transformer architecture Dosovitskiy et al. (2020) and data flow.

The applications of ViTs also go beyond state-of-the-art tasks in traditional image processing to the processing of LiDAR data by working on the segmentation of point clouds in autonomous driving. Using the implicit capability of the ViT models in handling sequence data for the capture of long-range dependencies, attempts have been made to enhance this interpretation of rich 3D LiDAR data. Moreover, due to the inherent capability of handling sequences in ViTs, temporal information can be added between time-consecutive frames to further enhance the segmentation with contextual understanding over time. Although this opens promising new research avenues for the application of ViTs to segmentation of

LiDAR data, their full potential, together with overcoming several challenges in their implementation, is yet to be explored.

4.2.3 Application of ViTs to LiDAR Segmentation

While the capability of a ViT holds promising direction, applications concerning segmentation in LiDAR data are still at their infancy. Initial works, such as Zhao, Jiang, Jia, Torr & Koltun (2021), started adapting the ViT architecture for the processing of point cloud data and achieved encouraging results. Usage of ViTs effectively for real-time LiDAR segmentation, concerning applications in autonomous vehicles is still an open research area and requires further investigation. In this chapter, we propose a new method for point cloud segmentation using only ViTs, following a spherical projection of the data.

4.3 Methodology

The rapid developments of the ViT network brought not only its performance above and beyond traditional CNNs but did so also for many deep learning applications related to the field of autonomous driving, such as object detection and segmentation. Drawing inspiration from these advances, our approach focuses on the use of only ViTs for the efficient and accurate segmentation of LiDAR point clouds, bypassing the reliance on conventional CNN architecture. We instead utilise a multitask learning framework that leverages multiple ViTs in our proposed methodology. Our approach enables the model to simultaneously learn feature representations and segment point cloud data, harnessing the transformers' capability to process sequential data and capture long-range dependencies within the inputs.

4.3.1 Spherical Projection

In contrast, working directly with raw 3D point cloud data is very challenging due to its high-dimensionality and unstructured nature. We use a spherical projection technique to make the input 2D data Inan et al. (2023). The dimensional reduction and added structure of the transformed data make it easier to process by transformer networks. A number of LiDAR sensors, including but not limited to Velodyne, produce data in a range image-like format where the columns are distances measured by an array of laser range-finders at one instant in time, while the rows correspond to different rotational positions of the same range-finders that are activated at regular intervals. For moving vehicles itself, you can cheerfully remark that some distortions may be introduced by the so-called "rolling shutter" effect since the sensor does not rotate within a very short time, let's say instantaneously. To account for this continuous motion of the vehicle in each scan, the result is compensated by the motion of it during the calculation phase. As a result, this point cloud does not capture the range measurement from every pixel of the camera uniformly; multiple measurements are recorded by some pixels.

Our methodology begins with transforming each de-skewed point cloud into a range-based representation. This is achieved by mapping each point $p_i = (x, y, z)$ through a function $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$, converting it to spherical coordinates, and subsequently to image coordinates. This process is formalized as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right] w \\ \left[1 - \left(\arcsin\left(\frac{z}{r}\right) + f_{\text{up}}\right) f^{-1}\right] h \end{pmatrix}, \tag{4.1}$$

where (u, v) represent the image coordinates, and (h, w) denote the height and width of the projected range image. Here, f is the sum of f_{up} and f_{down}, which correspond to the sensor's vertical field of view. Furthermore, r denotes the range

of each point, computed as $r = \|p_i\|$. This procedure yields a set of (u, v) pairs, where each pair indicates the image coordinates for a specific point p_i. For each point p_i, we extract its range r, coordinates x, y, z, and remission, forming a $[5 \times h \times w]$ tensor.

4.3.2 Network Architecture

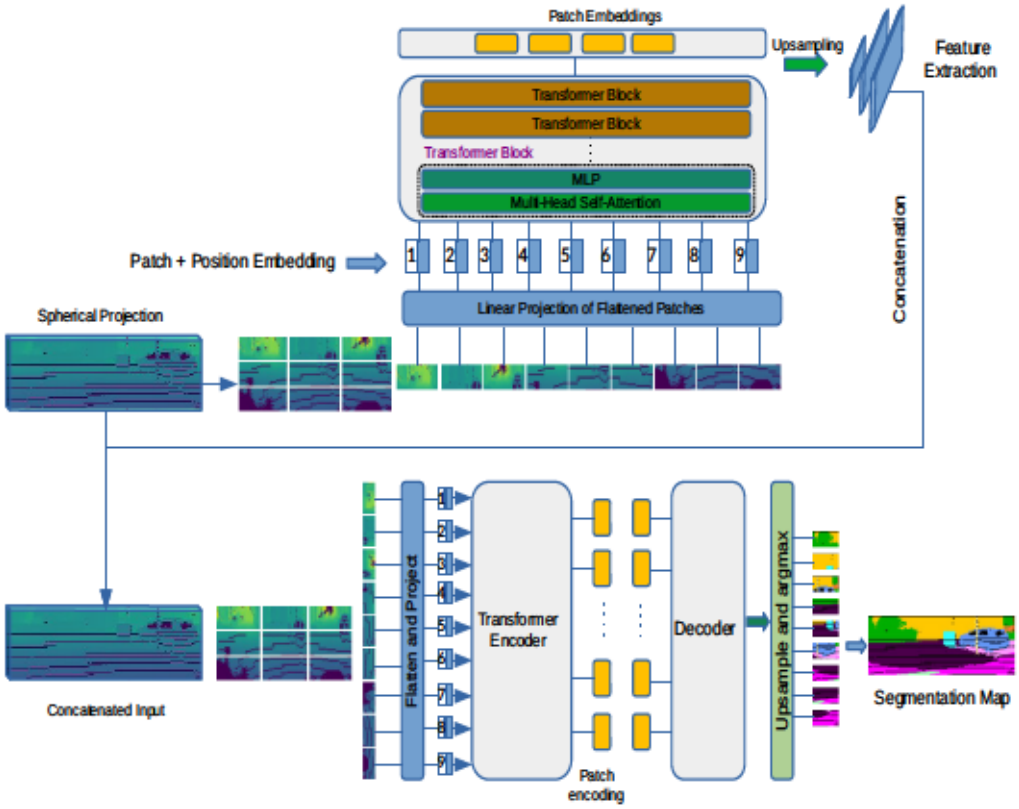

Figure 4.3: Our model architecture. ViT used for feature extraction and these features later used as additional input to the second ViT which is responsible for segmentation.

Our architecture is designed as a two-stage transformer-based framework for LiDAR point cloud segmentation, combining advanced feature extraction and segmentation into a cohesive pipeline. The workflow leverages two distinct ViTs, enabling it to capture both global contextual information and localized details essential for accurate segmentation.

The process begins with spherical projection pre-processing, where the raw LiDAR point cloud data is transformed into a pseudo-image representation. This transformation enables the ViT to operate efficiently by treating the input as a grid of non-overlapping patches. Each patch is embedded into a fixed-dimensional vector through a linear projection layer. This patch embedding step allows the model to extract meaningful local features while preserving spatial relationships within the input data.

The embedded patches are then processed by the first ViT using self-attention mechanisms, which capture both short-range dependencies and long-range contextual relationships between patches. This capability is crucial for understanding complex spatial patterns and interactions present in the LiDAR data. The output of the first ViT is a set of high-level feature representations that encapsulate both local and global information, which serve as the foundation for subsequent segmentation tasks.

To bridge the gap between the patch-based embeddings generated by the first ViT and the per-point resolution required for segmentation, we employ a transposed convolution layer. This operation upscales the feature map to match the spatial resolution of the original input data, ensuring that each point in the point cloud is associated with a dense feature vector. By combining the localized patch information with the global context provided by the transformer, this step produces a rich and detailed feature map, which is critical for precise segmentation.

The second ViT takes the upsampled feature map as input and is explicitly

trained for segmentation. It predicts per-point segmentation labels by processing the dense feature vectors through its self-attention layers. These layers refine the features by modeling complex dependencies between points, enabling the network to handle challenging scenarios such as sparse regions, occlusions, and overlapping objects.

A key aspect of this stage is the integration of the feature representations generated by the first ViT. By using the outputs of the feature extraction stage as additional input, the second transformer learns to associate these high-level features with the specific requirements of the segmentation task. This design ensures that the segmentation predictions are both accurate and consistent with the underlying features of the input data.

The entire architecture is trained end-to-end, leveraging the strengths of multitask learning to optimize feature extraction and segmentation simultaneously. The training process uses raw LiDAR point cloud data paired with ground truth segmentation labels. During training, the model learns to minimize a combination of loss functions designed to align feature representations with segmentation objectives. We present an overview of the overall architecture of our approach in Figure 4.3.

The segmentation task employs a cross-entropy loss to measure the discrepancy between predicted and ground truth labels. The model is trained using the AdamW optimizer, with a learning rate schedule based on cosine annealing. This schedule gradually reduces the learning rate during training, facilitating fine-tuning in the later stages and preventing overfitting. Dropout layers are employed within the transformers to further mitigate overfitting, while weight decay regularizes the model's parameters.

To ensure efficient training, we use a batch size optimized for the computational resources available, balancing convergence speed with memory require-

ments. The dual-transformer design allows the model to share information between the feature extraction and segmentation stages, creating a feedback loop that enhances learning across both tasks. The end-to-end training process ensures that the feature representations learned by the first transformer are not only generalizable but also tailored to the specific requirements of segmentation.

This dual-transformer architecture combines the strengths of ViT in capturing both global and local context with an efficient training process that aligns feature extraction and segmentation. By leveraging spherical projection, transposed convolution, and multitask learning, our approach achieves robust and accurate segmentation of LiDAR point clouds.

Loss Function

Many neural networks suffer from this problem, where one class or another within a dataset can be highly imbalanced. Consider autonomous driving; an object like a bicycle or a traffic sign may appear much more rarely compared to vehicles. This can easily result in biasing the network to classes that are more frequent within the training data, because the performance on infrequent classes becomes significantly poorer. To address this, the network is trained end-to-end using stochastic gradient descent and a weighted cross-entropy loss function $\mathcal{L}$:

$$\mathcal{L} = -\sum_{i=1}^{C} w_i y_i \log(\hat{y}_i), \tag{4.2}$$

$$\text{where} \quad w_i = \frac{1}{\log(f_i + \epsilon)} \tag{4.3}$$

This loss function applies a weight w_i to each class i, computed as the inverse of the logarithm of its frequency f_i. By doing so his loss helps in tackling the problem of data imbalance, which is typical in tasks like semantic segmentation,

in which classes like "road" could have substantially more data points compared to classes like "pedestrian".

4.4 Results

After all, our approach was trained and evaluated on the publicly available dataset, containing dense, point-wise annotations for the complete KITTI Odometry Benchmark specified by Geiger et al. (2012) and Behley et al. (2019). The dataset includes over 43,000 LiDAR scans. Of these, over 21,000 scans of sequences 00 to 10, excluding sequence 08, were used for training. The remaining scans of sequences 11 to 21 were held out for testing purposes. Sequence 08 was set aside solely as a validation set to support choosing the best hyperparameters while training the model; doing so gives an assurance that the model is well trained and its performance validated against representative samples from the diversified KITTI dataset. We compare our model's results with state-of-the-art results in Table 4.1.

Their dataset features a total of 22 unique classes, 19 of which were tested on a test set using the SemanticKITTI benchmark website. The wide-ranging nature of this evaluation makes sure that our model's performance, in regard to all these categories, is well-checked for a proper understanding of its strengths and weaknesses.

We used different visualizations while assessing the semantic segmentation performance by comprehensively analyzing how effective the performance of our model was due to the difference in classes and scenarios.

Figure 4.4 shows the grouped bar plot of the IoU scores of different semantic classes obtained for each of the methods evaluated. In such a plot, a direct comparison among the model performances on specific object categories is easy to perform. It highlights how methods like 2DPASS and RangeNet53++ perform ex-

Table 4.1: Semantic segmentation results on the SemanticKITTI test benchmark (sequences 11 to 21).

Method	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Pointnet Qi, Su, Mo & Guibas (2017)	14.6	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7
Pointnet++ Qi, Yi, Su & Guibas (2017)	20.1	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9
SPGraph Landrieu & Simonovsky (2018)	20.0	68.3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5
SPLATNet Su et al. (2018)	22.8	66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0
TangentConv Tatarchenko et al. (2018)	22.8	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2
SqueezeSeg Wu et al. (2018)	29.5	68.8	16.0	4.1	3.3	3.6	12.9	13.1	0.9	85.4	26.9	54.3	4.5	57.4	29.0	60.0	24.3	53.7	17.5	24.5
SqueezeSeg-CRF Wu et al. (2018)	30.8	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3
SqueezeSegV2 Wu et al. (2019)	39.7	81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3
SqueezeSegV2-CRF Wu et al. (2019)	39.6	82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0
PointASNL Yan et al. (2020)	46.8	87.9	0.0	25.1	39.0	29.2	34.2	57.6	0.0	87.4	24.3	74.3	1.8	83.1	43.9	84.1	52.2	70.6	57.8	36.9
RangeNet21 Milioto et al. (2019)	47.4	85.4	26.2	26.5	18.6	15.6	31.8	33.6	4.0	91.4	57.0	74.0	26.4	81.9	52.3	77.6	48.4	63.6	36.0	50.0
RangeNet53++ Milioto et al. (2019)	52.2	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
PolarNet Zhang, Zhou, David, Yue, Xi, Gong & Foroosh (2020)	54.3	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	90.0	61.3	84.0	65.5	67.8	51.8	57.5
RandLA-Net++ Hu et al. (2020)	55.9	94.2	29.8	32.2	43.9	39.1	48.4	47.4	9.4	90.5	61.8	74.0	24.5	89.7	60.4	83.8	63.8	68.6	51.0	50.7
Cylinder3D Zhu et al. (2021)	68.9	97.1	67.6	63.8	50.8	58.5	73.7	69.2	48.0	92.2	65.0	77.0	32.3	90.7	66.5	85.6	72.5	69.8	62.4	66.6
2DPASS Yan et al. (2022)	72.9	97.0	63.6	63.4	61.1	61.5	77.9	81.3	74.1	89.7	67.4	84.7	40.0	93.5	72.9	86.2	73.9	71.0	65.0	70.4
Ours	41.2	83.6	19.2	17.4	19.5	16.5	18.7	23.6	3.4	87.4	48.7	67.4	15.6	76.1	42.3	79.4	35.1	61.4	28.9	38.2

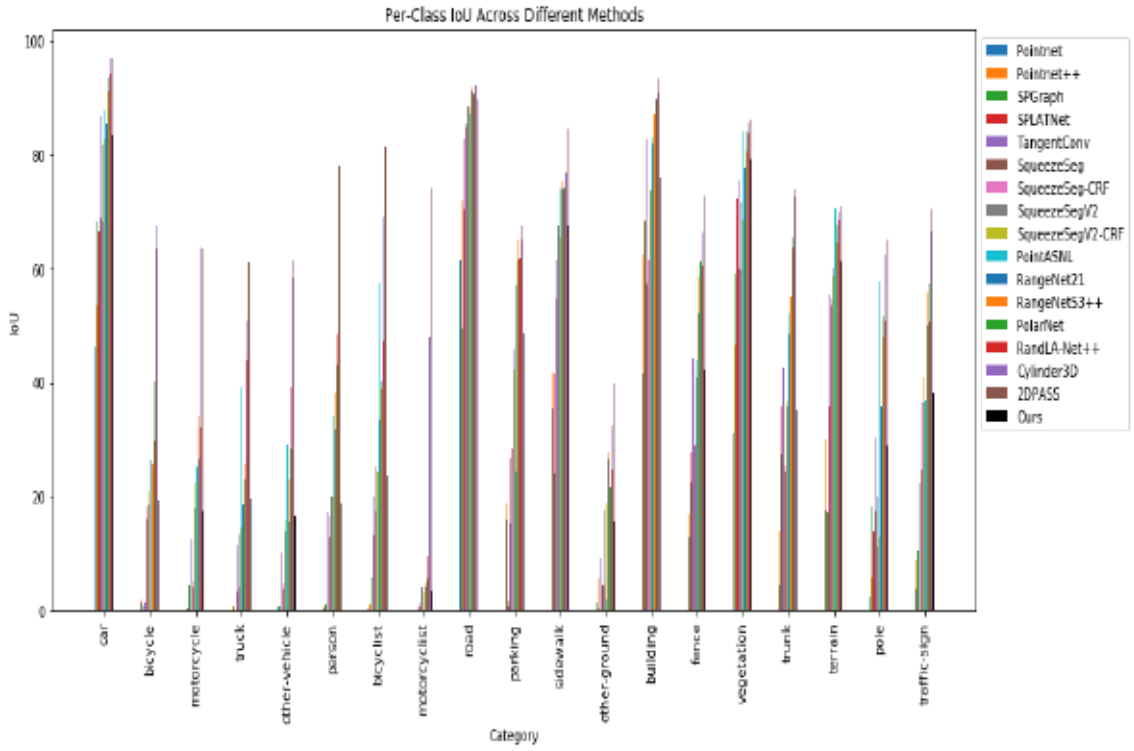

Figure 4.4: Grouped bar plot of the IoU scores for the classes.

ceptionally well in classes such as 'car' and 'road', but less so in underrepresented categories like 'bicycle' and 'motorcycle'.

While the grouped bar plot gives a very clear comparison of class-wise performance across methods, it is equally useful to investigate the consistency of these methods across all classes. We further provide a heatmap visualization that serves this purpose for further investigation.

Figure 4.5 shows the IoU score heat map per class and method. This provides a very interesting overview of the segmentation performances of all methods on this dataset. It may be immediately clear from the heat map which of the methods have consistent performance over classes, and which methods may have more variability within their performance. In particular, Cylinder3D and 2DPASS perform good over a large number of classes, which means that they can generalize

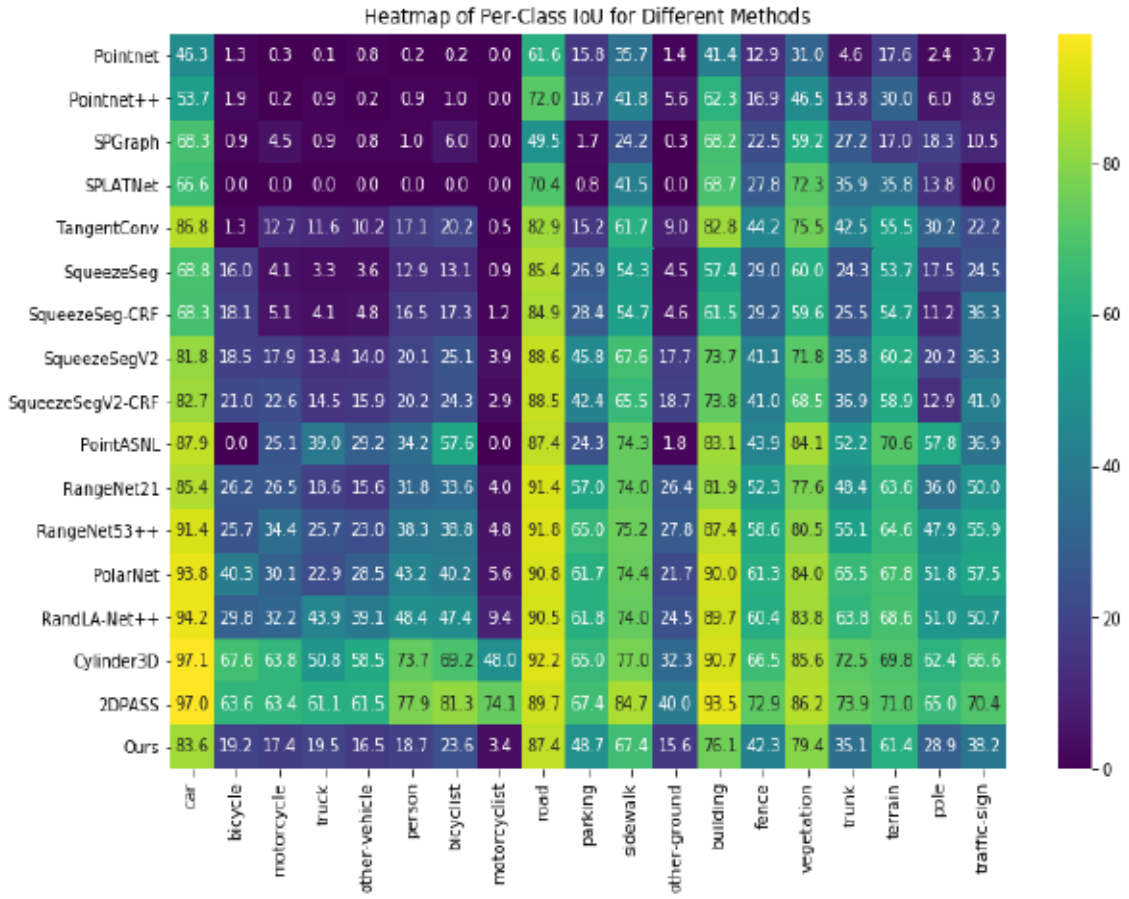

Figure 4.5: Heatmap per class.

to various object types.

Further clarification comes from the stacked bar plot, where it provides a detailed analysis of how different classes contribute to the overall performance of every method.

Figure 4.6 depicts a stacked bar plot showing the contribution of each class to the overall IoU in the case of various methods. It is useful to understand the reflection of each class in the total performance. Methods such as Cylinder3D and 2DPASS depict equal variance in the contributions of diverse classes, indicating the flexible ability to deal with all object types compared to other methods which

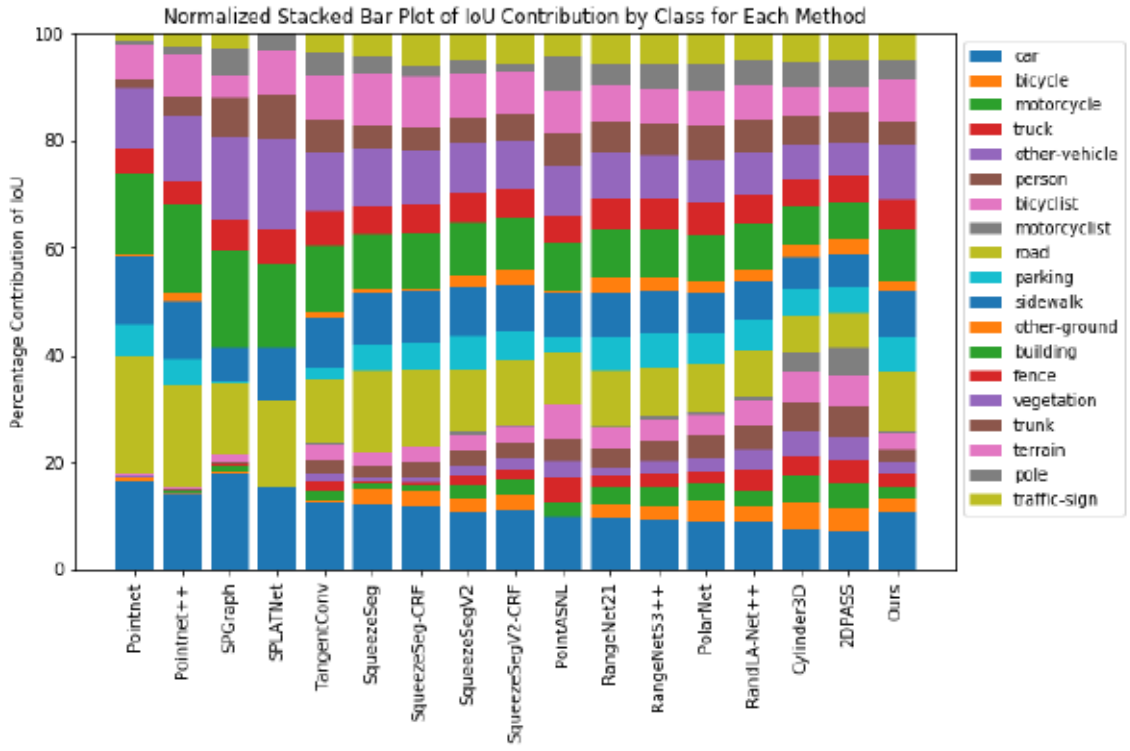

Figure 4.6: Stacked bar plot for the contributions of each class to the IoU.

depend on high performance classes such as 'car' and 'road'.

Knowing the distribution of instances among the classes in the dataset provides important context for interpreting the segmentation results. Figure 4.7 shows distribution of classes in the dataset. It shows very heavy imbalance that exists in this dataset. There is heavy dominance within the dataset of 'car' and 'road' classes, with very few representations of 'bicycle' and 'motorcycle' classes. This imbalance impacts model training and evaluation, and understanding it will be helpful in interpreting the performance metrics and thereby guiding improvements to model design.

These visualizations let us intuitively gauge performance trends across methods and by different dataset characteristics. Together, they provide a comprehensive overview of the segmentation performance.

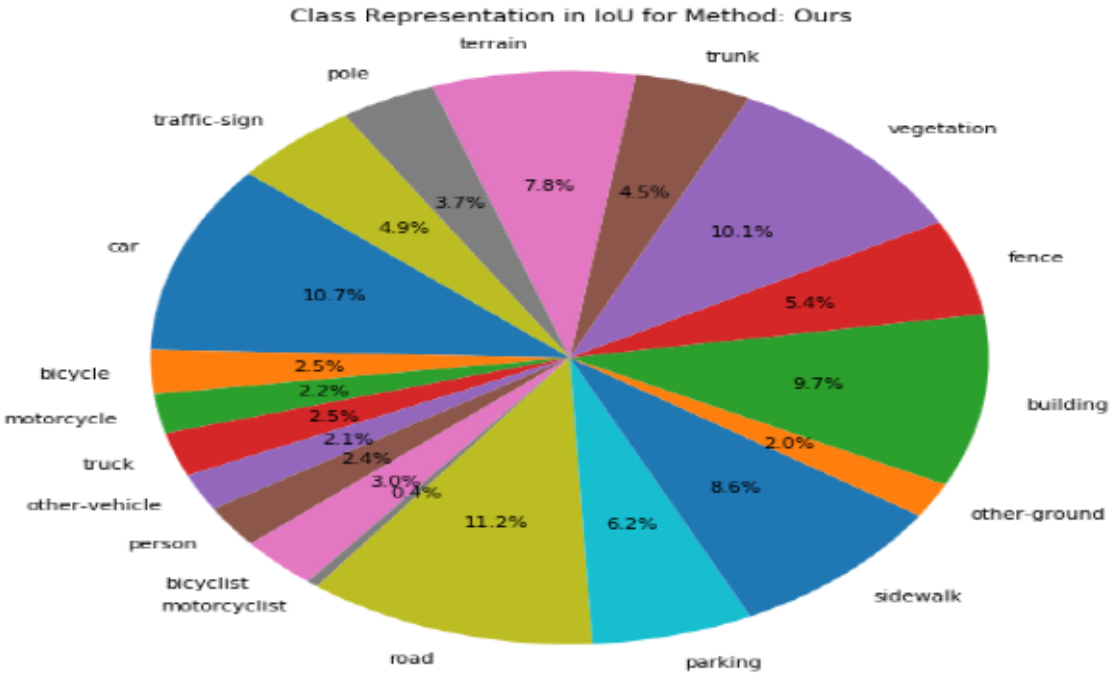

Figure 4.7: Piechart per class shows the distribution.

4.4.1 Hyperparameter Tuning

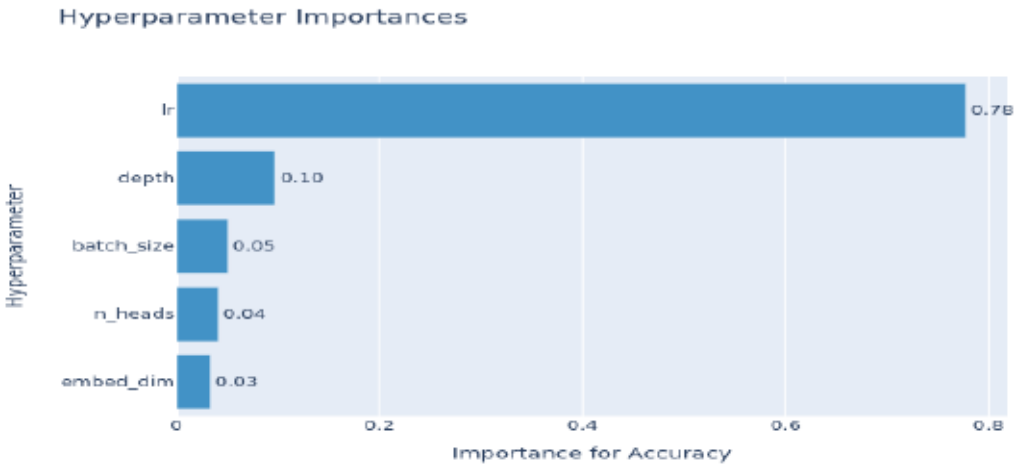

Figure 4.8: Importances plot for hyperparameter selection process.

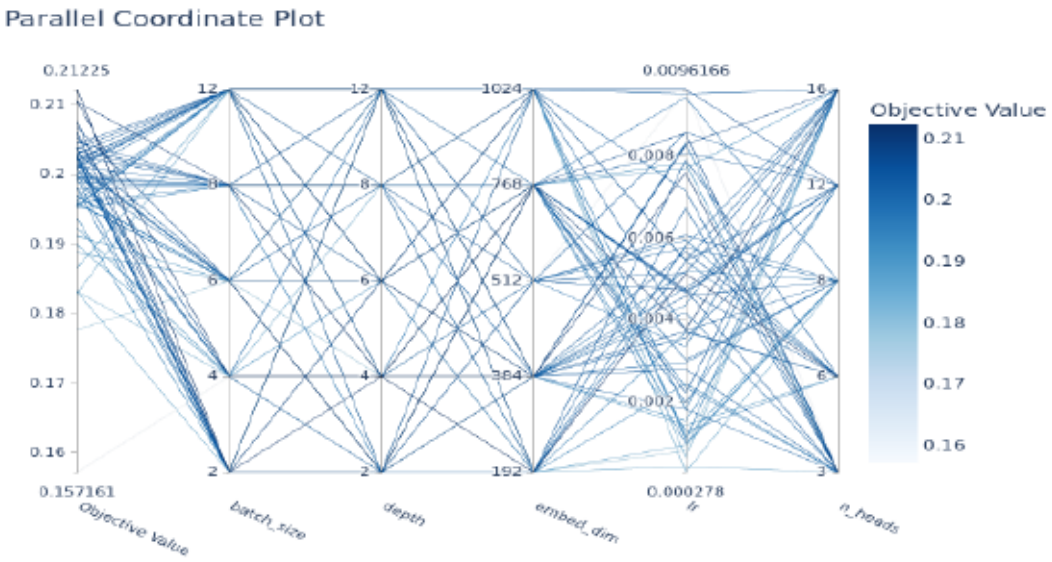

Figure 4.9: Parallel coordinate plot for hyperparameter optimization.

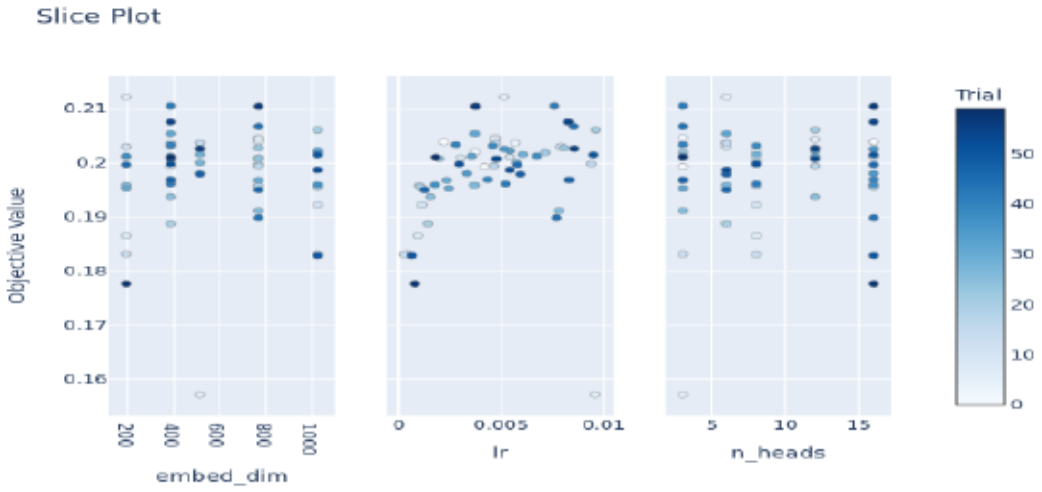

Figure 4.10: Slice plot for hyperparameter optimization.

We illustrate hyperparameter tuning process via several different visuals. The importance plot 4.8 extends beyond mere ranking; it highlights which hyperparameters are most impactful, thereby aiding in prioritizing those that require fine-

tuning to achieve optimal performance. The parallel coordinate plot 4.9 further explores how hyperparameters interact with each other and their combined effect on the objective value. The slice plot 4.10 offers a detailed view of the influence of individual hyperparameters on the objective value, allowing us to discern the sensitivity of the model to each parameter's adjustments. This is especially necessary for understanding the marginal effect of each hyperparameter on overall performance. Optuna library was utilised for this detailed analysis which gives us complex relationships, synergies, and potential conflicts between parameters, providing valuable insights for refining the model. After the analysis, the optimal learning rate value was determined, and a decay factor of 0.99 was applied for each subsequent epoch, and the model trained over 150 epochs.

4.4.2 Evaluation Metrics

To quantitatively assess the performance of our model in semantic segmentation, we employed the mean Intersection-over-Union (mIoU) metric, commonly referred to as mIoU. This metric, detailed in Equation 4.4 Everingham et al. (2015), evaluates the average overlap between predicted and true regions across all classes, providing a holistic measure of the model's classification accuracy. The calculation of mIoU considers true positive (TP_c), false positive (FP_c), and false negative (FN_c) predictions for each class c, as formulated below:

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FP_c + FN_c} \tag{4.4}$$

Here, C denotes the total number of distinct classes in the dataset. By encompassing all classes, this metric provides a robust evaluation of the model's ability to correctly classify instances across a wide range of categories. The detailed results

are presented in Table 4.1.

4.4.3 Qualitative Results

In this context of an experimental setup, we have explored how well a purely ViT-based architecture can perform in segmenting LiDAR point clouds without the support of a CNN backbone or hybrid CNN-ViT models.

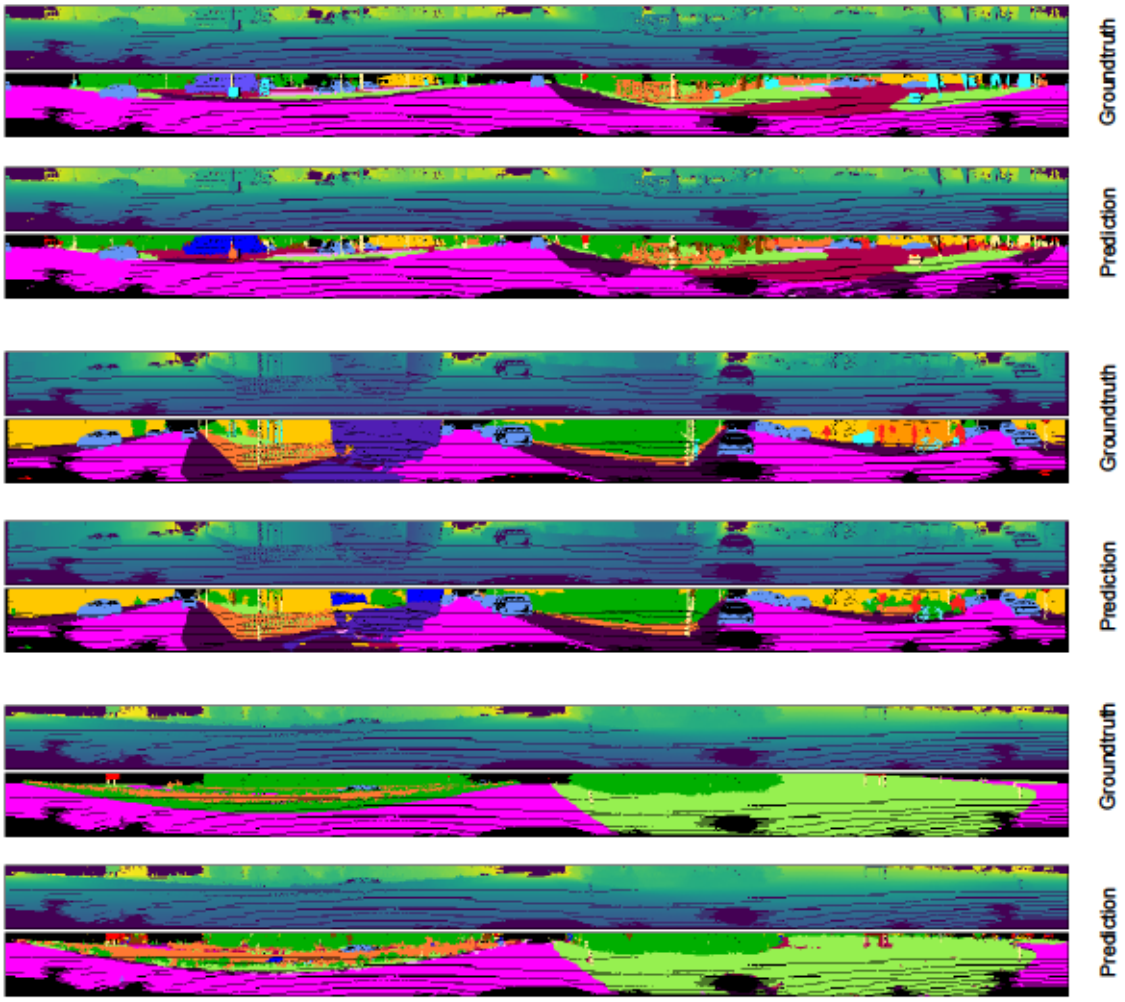

Figure 4.11: Qualitative results of the network.

As illustrated in Table 4.1, our approach outperforms several previous works based on CNNs, without any use of a pretrained ViT. Nevertheless, it does not yet

achieve the performance of the advanced state-of-the-art models, such as those proposed by Zhu et al. (2021) and Yan et al. (2022). This might be due to the fact

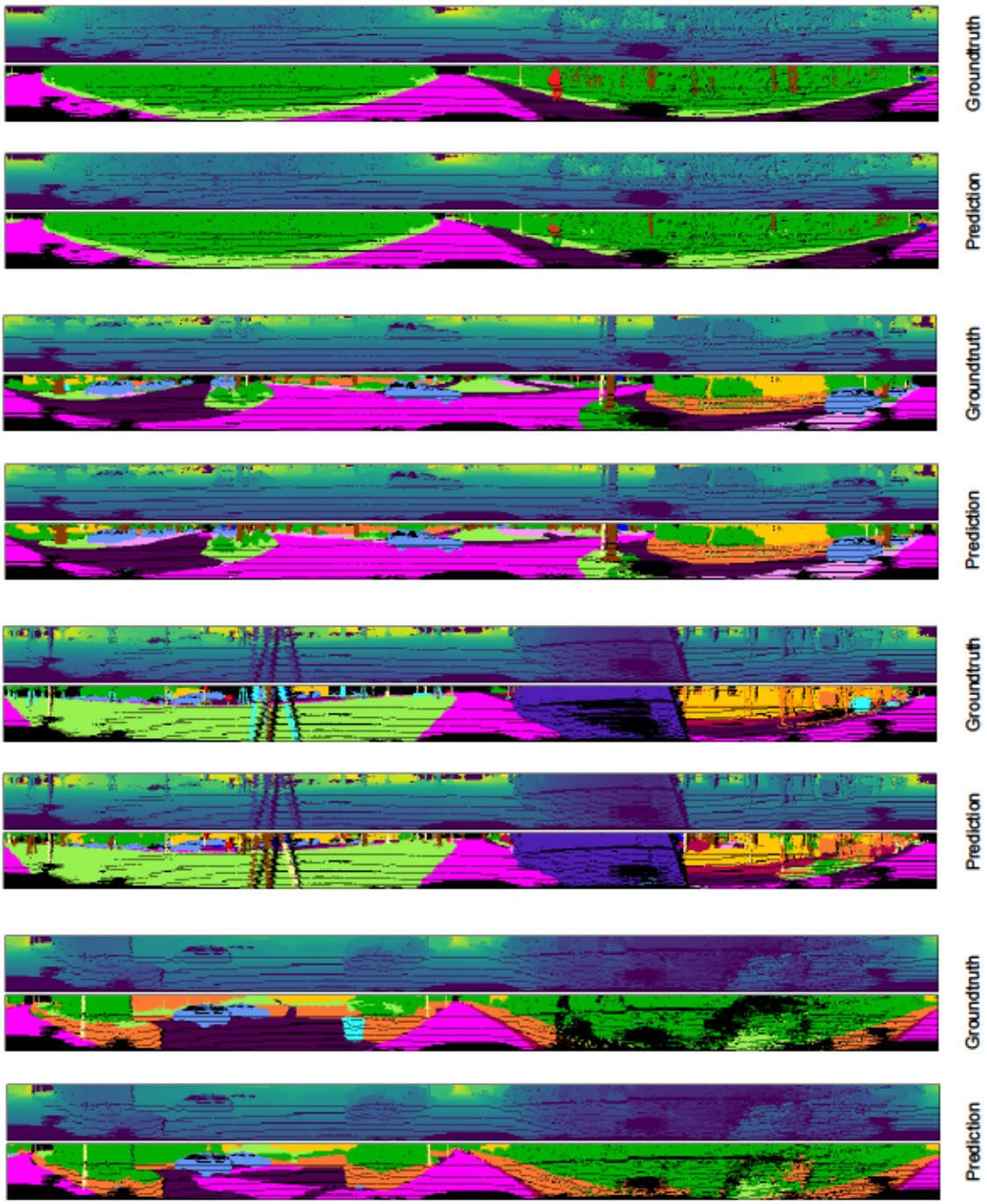

Figure 4.12: Qualitative results.

Figure 4.13: Training loss per number of epochs during the process.

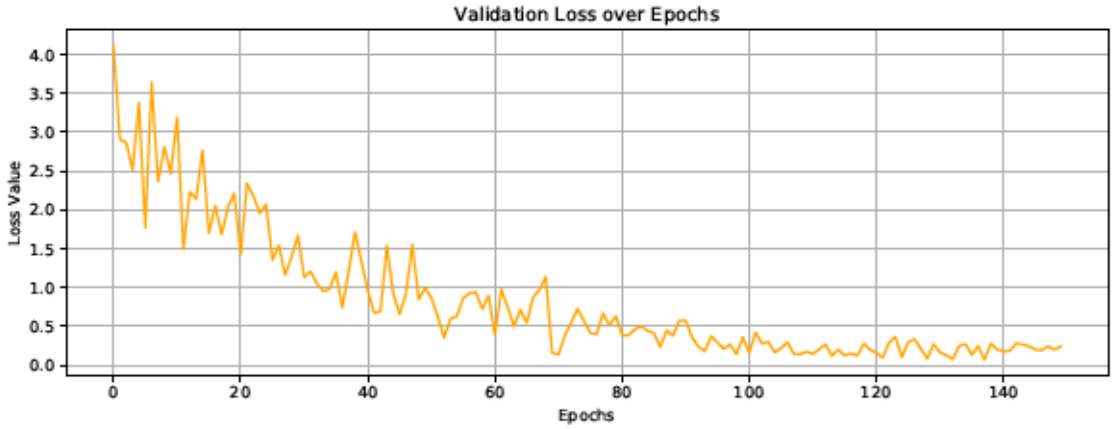

Figure 4.14: Validation loss per number of epochs during the process.

that intrinsically, large datasets are the grounds for the preference of ViTs. Furthermore, our decision to process $64 \times 2048 \times 5$ inputs directly after projection without initially using a CNN to reduce the feature map could contribute to these results. To mitigate the absence of a pretrained network, we processed feature maps from the first ViT through a second ViT, acknowledging opportunities for further optimization. Our findings underscore the potential and flexibility of standalone ViT models in this domain, suggesting promising directions for future research and

development. Some qualitative results are given in Figure 4.11 and Figure 4.12. Groundtruth images and the corresponding predicted images are shared together. The loss values during the training and validation can be seen in Figure 4.13 and Figure 4.14.

4.5 Conclusion

In this chapter, we proposed a novel method for LiDAR point cloud segmentation, specifically addressing the challenges unique to autonomous driving. Distinguishing itself from traditional approaches that predominantly rely on CNNs, our method leverages ViTs to exploit their abilityin capturing the long-range dependencies and modeling complex spatial relationships inherent in point cloud data.

Our architecture is built on a dual-transformer model design. The first model extracts meaningful feature maps from raw LiDAR data, and the second transformer uses these feature maps for segmentation. By integrating two distinct ViT-based models, the approach enhances the understanding and processing of spatial and contextual information, improving segmentation quality. This layered design exemplifies the versatility of transformers in autonomous driving tasks.

A notable feature of our approach is its independence from pre-trained models, which are often essential for improving performance in related tasks. Training the transformers from scratch demonstrates their potential to achieve competitive results on standard benchmarks without relying on pre-existing weights. This independence simplifies implementation and opens avenues for further exploration and optimization.

While the results are encouraging and demonstrate that ViTs can serve as a competitive alternative to CNNs for LiDAR point cloud segmentation, there remains room for refinement and enhancement. Further research could explore

augmenting transformer architectures or integrating additional sensor modalities to enhance segmentation performance in diverse real-world scenarios.

The findings of this chapter have broader implications for the perception systems of autonomous vehicles. By demonstrating the feasibility and promise of using transformers for segmentation, this work sets the stage for exploring their application to other autonomous driving tasks. Specifically, the insights gained here directly influenced the direction of the subsequent chapter, which delves into object tracking using transformers. The need to model complex spatial interactions and temporal dependencies identified in this chapter motivated the adoption of transformer-based models for multi-object tracking. Additionally, the independence from pre-trained models explored here highlights the potential for customized architectures tailored to specific challenges in object tracking and motion forecasting.

In summary, this chapter shows the feasibility and promise in rethinking traditional methodologies for LiDAR point cloud segmentation. By exploring the capabilities of ViTs and challenging the reliance on CNNs, this work contributes to the ongoing effort to develop innovative, robust, and effective autonomous systems. It provides a strong foundation for the trajectory of this thesis, as the methodologies and findings inspire and inform subsequent research on perception and prediction tasks in autonomous driving.

CHAPTER 5

Enhanced Multi-Object Tracking Based on Transformers and Sensor Fusion

This chapter will introduce a new framework of multi-object tracking using a transformer-based architecture, combined with sensor fusion, in order to achieve higher accuracy in dynamic environments. The methodology will therefore incorporate both 2D and 3D object detection, fusing data from both LiDAR and cameras for better interpretation of the traffic agents' movement. Sensor fusion will allow the tracking system to estimate, on every frame in runtime, both the exact position and velocity of the moving agents. In the proposed tracking framework, transformer networks use advanced attention mechanisms to capture both local interactions and global contexts. An evaluation of the system performance is presented in this chapter to demonstrate how effectively challenges for complex tracking scenarios in an urban driving environment can be handled.

5.1 Motivation

In the broadening area of autonomous driving and mobile robotics, safety and efficiency in vehicle navigation are of great concern. The capability for clear perception and the forecasting of motion for objects in proximity is very critical; thus, systems should avoid accidents or navigate through challenging environments. This calls for not only object detection and segmentation, but with the precise tracking object trajectories in 3D space both in the time and spatial domains.

One of the central issues of concern within this scope is the so-called 3D object detection task, which needs to identify objects both in terms of their spatial location and category. Approaches such as VoxelNet Zhou & Tuzel (2018) and PointPillar Lang et al. (2019) have utilised LiDAR sensors for creating point clouds, thus obtaining valuable three-dimensional data. But, these methods perform poorly on sparse point clouds, in particular on large-scale datasets such as nuScenes Caesar et al. (2020) and Waymo Sun et al. (2020), the detection of small or far objects becomes quite a challenging task. High-resolution cameras capture finer details of objects at a longer distance. This would include details that can provide complementary information useful for enhancing the overall perception system.

Multi-object tracking extends the capability of any detection system to monitor the trajectory of multiple objects continuously, preserving their identity between consecutive frames. In self-driving vehicles, 3D MOT allows us to have a nuanced situational awareness of dynamic environments and decision-making processes. Tracking systems based on LiDAR have indeed been performing well so far due to their ability to provide precise 3D information, which is needed for object tracking to be very accurate. However, in reality, these systems have faced severe limitations from being sensitive to reflective surfaces and signal sparsity, that can reduce

their performance. On the contrary, image-based approaches present strong visual information; even partial occlusion does not affect them, and excellent localisation within the image plane is achieved Voigtlaender et al. (2019), Sharma et al. (2018).

Although such advances have been made, there is still a great gap in how 2D and 3D sensor data are integrated to enhance accuracy and robustness of tracking in real-time applications. Most of those methods would usually fail due to complex occlusions when object interactions are normal in certain cluttered environments. Such challenges therefore call for innovative methods which could fuse such multi-modal data and adapt dynamically to the evolving scenarios.

An enhanced framework for vehicle tracking, which involves transformer-based object detection and sensor fusion methods to tackle the aforementioned issues is presented in this chapter.Utilising a novel transformer architecture that integrates 2D and 3D sensor data, the target of the method is to achieve substantial improvement in accuracy and robustness. It consists of a combination of three key components: transformer-based 2D and 3D object detection, sensor data fusion and association, and transformer-based track management. The effectiveness of the followed method is proven by extensive evaluations on nuScenes and KITTI. Our proposed method has great potential for further enhancements in various autonomous systems operating within complex real-world environments.

5.2 Related Work

5.2.1 Vision-Based 2D Multi-Object Tracking (MOT)

This field of vision-based 2D MOT keeps on developing with continuous object detection technology improvement. The early methods in their core relied on the

advancement of CNN for the detection and identity consistency of objects across frames. For example, TrackR-CNN Voigtlaender et al. (2019) extends Mask R-CNN He et al. (2017) with combining 3D convolutional networks to improve temporal consistency and object reidentification techniques to make the object association across frames more robust. Tracktor Bergmann et al. (2019) revised the head of Faster R-CNN Ren et al. (2015) to predict positions of objects over time correctly from the regression head. CenterTrack Zhou, Koltun & Krähenbühl (2020) proposed an offset-regression head for associating objects across frames, hence improving detection continuity Zhou et al. (2019).

Recent works in this area have moved to end-to-end learning-based models Xu, Osep, Ban, Horaud, Leal-Taixé & Alameda-Pineda (2020), Frossard & Urtasun (2018), using graph neural networks (GNNs) to learn more sophisticated association strategies Brasó & Leal-Taixé (2020),Weng, Wang, Man & Kitani (2020). These contributions indicate a general tendency of integration of so many components related to the tracking pipeline for better but also efficient tracking systems. Most of such methods often encounter obstacles like occlusions, fast motion, and appearance changes that make the continuity of tracking broken with reduced accuracy. In order to address these problems, we develop a strategy that ensures the exploitation of the global context provided by transformers. Thus, we propose the use of transformers for both detection and association to enhance robustness in tracking, especially complicated interaction and partial occlusion scenes.

5.2.2 LiDAR-Based 3D Multi-Object Tracking (MOT)

Different LiDAR-based 3D multi-object tracking methods have been progressively developed with the advance of autonomous driving technologies. Initial approaches focused on segmenting LiDAR scans and performing segment association across

frames to track objects over time Teichman et al. (2011), Moosmann & Stiller (2013). The emergence of point cloud representation learning techniques, including PointNet Qi, Su, Mo & Guibas (2017), PointNet++ Qi, Yi, Su & Guibas (2017), together with the advance of 3D object detection Chen et al. (2015), Shi et al. (2019), Shi & Rajkumar (2020), led to a shift in the research direction toward tracking-by-detection methods with LiDAR and stereo data Osep et al. (2017), Frossard & Urtasun (2018). These have considerably boosted the accuracy of 3D object tracking owing to the precise point cloud data detecting and tracking objects in three-dimensional space.

Despite these developments, LiDAR-based tracking approaches fail to evade a number of challenges. As one example, reliance on 3D detections makes methods prone to false positives and occlusion gaps, which can impact tracking performance, especially within dynamic environments presenting high object density Weng, Wang, Held & Kitani (2020). Recent approaches have tried to bridge this gap. For example, CenterPoint Yin et al. (2021) detects 3D object centers and establishes frame-to-frame associations based on predicted velocity vectors. The challenges are that it is difficult to maintain consistent tracks and handle occlusions accurately.

Our approach improves how 3D detectors are combined with transformer-based models by stimulating robust feature fusion and object association in 3D tracking. This, therefore, lets the result retain higher tracking coherency and accuracy during difficult situations.

5.2.3 Tracking Paradigms

In MOT, several paradigms have been devised to handle complexities arising in object association between frames. These can broadly be categorized as follows:

Tracking-by-Detection

In the tracking-by-detection paradigm, object trajectories are created by associating detections over time. This has been widely adopted since it is simple to implement and works well in a number of cases. Approaches belonging to this paradigm usually rely on graph-based methods for track association and long-term re-identification, using algorithms such as maximum flow (minimum cost) optimisation Berclaz et al. (2011), and distance-based association Pirsiavash et al. (2011). Other approaches incorporate motion information Keuper et al. (2018) and trainable graph neural networks Brasó & Leal-Taixé (2020), Wang, Kitani & Weng (2021) to achieve better association performance. Unfortunately, the computational complexity brought in by graph-based methods often makes them unpractical for deployment in real-time applications.

Motion-Based Tracking

The motion-based tracking methods estimate the object trajectories under the assumed motion model, such as a constant velocity Andriyenko & Schindler (2011). These models are very helpful for track association across frames, by predicting future object positions from previously estimated motion Zhang, Sheng, Wu, Wang, Lyu, Ke & Xiong (2020). However, these methods have difficulty in correctly modeling nonlinear 3D motion and its projection into a 2D image domain, where the amount of visual information may be sparse.

Tracking-by-Regression

Tracking-by-regression methods break the conventional association of detections, as they regress the object's past locations to predict its new position in the current frame. Usually, these methods apply regression heads to region-pooled object fea-

tures Bergmann et al. (2019), Feichtenhofer et al. (2017), or are expressed as center points under distance-based association Zhou, Koltun & Krähenbühl (2020). More often than not, additional improvements are done through re-identification models and graph-based approaches Brasó & Leal-Taixé (2020) for enhancing identity preservation and track reasoning.

Tracking-by-Segmentation

Tracking-by-segmentation approaches predict object masks and leverage pixel-level information in crowded scenes with ambiguous backgrounds. In the family of methods, category-agnostic segmentation Ošep et al. (2018) and Mask R-CNN He et al. (2017) using 3D convolutions Voigtlaender et al. (2019), have shown promise in densely packed scenes. However, as annotated MOT segmentation data is lacking, most recent works still recognize object tracking using bounding box annotations.

5.2.4 Transformers and Attention Mechanisms in Tracking

Recently, transformers and attention mechanisms attracted extensive interest due to their applications on various computer vision tasks. Attention mechanisms have been used in multi-object tracking for improving object detection and association Carion et al. (2020), Zhu et al. (2020). These allow for selective focusing by the models on different parts of the input. They capture, in a particularly effective way, dependencies across sequences and enhance robustness in tracking Chu et al. (2017), Zhu et al. (2018).

Building on this, our approach further uses transformers for the detection and tracking tasks. Transformers, first proposed by Vaswani et al. (2017), uses self-attention mechanisms that process sequences of tokens, each represented as a

feature vector. The attention mechanism computes the weights between tokens by taking a dot product between query and key vectors, respectively. This is an important part that is giving the model the capability to focus dynamically on diverse aspects of an input sequence. Using several attention heads in parallel, the model can gain the capabilities to capture diverse features and relationships in data toward comprehensive scene understanding.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{5.1}$$

where in Equation 5.1, d_k represents the dimension of the keys (K) and queries (Q). In cross-attention, query vectors are generated from a second sequence rather than from the same sequence as in self-attention. The encoder-decoder architecture allows the model to have both the self-attention and cross-attention mechanisms to take advantage of both spatial and temporal information toward tracking tasks. By harnessing these capabilities, our approach endeavors to conduct integrated, robust multi-object tracking in insightful environments.

5.3 Methodology

Our proposed method introduces an improved multi-object tracking framework that fuses 2D and 3D information with an enhanced transformer-based network. To sum up, this framework is designed in a way such that it comes up with some complementary strengths: 2D image data and 3D point cloud data, for enhancing accuracy and robustness in the capture of object tracking over dynamic environments. The system's architecture consists of three main components: Detection Transformer (DETR) for 2D object detection, Yin et al. (2021) for 3D object detection, and a novel integrated transformer module performing the feature fusion, data association, and tracking.

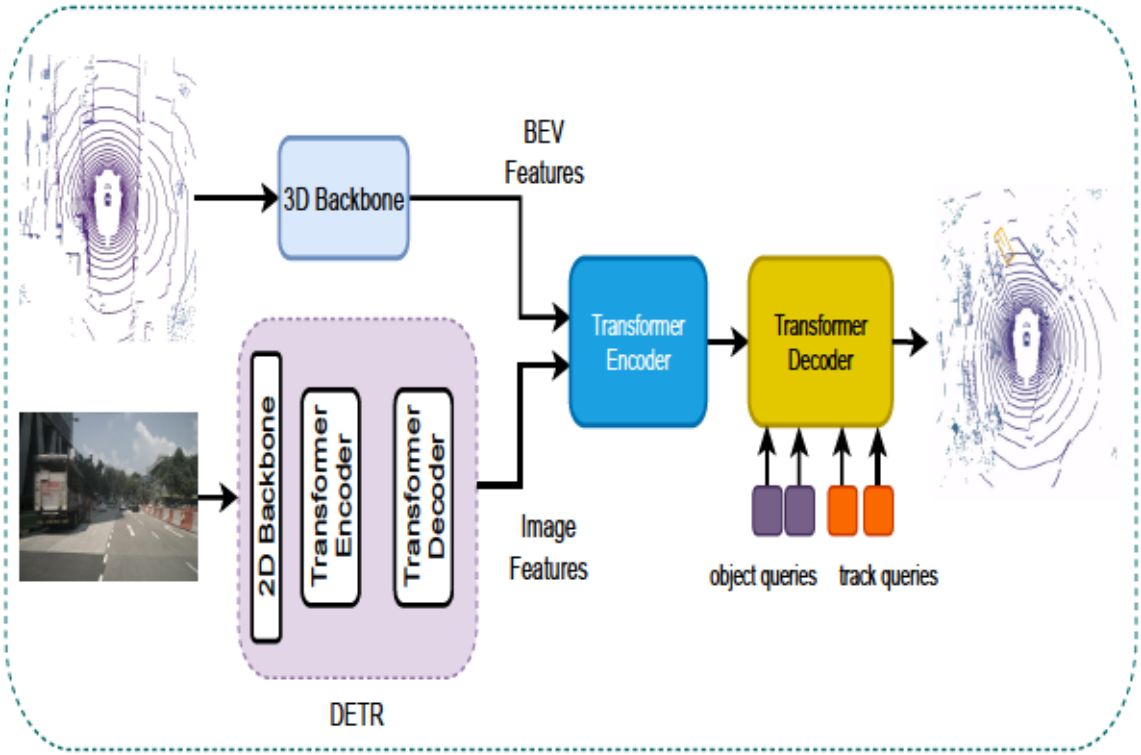

Figure 5.1: Comprehensive architecture of the proposed multi-object tracking framework. The framework includes DETR for 2D detection, Yin et al. (2021) for 3D detection, and a detailed unified transformer for fusion, association, and tracking. This transformer uses object and track queries to provide bounding boxes with IDs as output.

5.3.1 2D Object Detection with DETR

First, our framework incorporates DETR for 2D object detection. For the detection, a CNN backbone 5.2 is taken into consideration in order to achieve rich feature extraction from the input images. Further, extracted features are fed into the transformer encoder of DETR, which applies self-attention mechanisms in order to model relationships and spatial context within the image. While capturing long-range dependencies and contextual information, the encoder improves feature

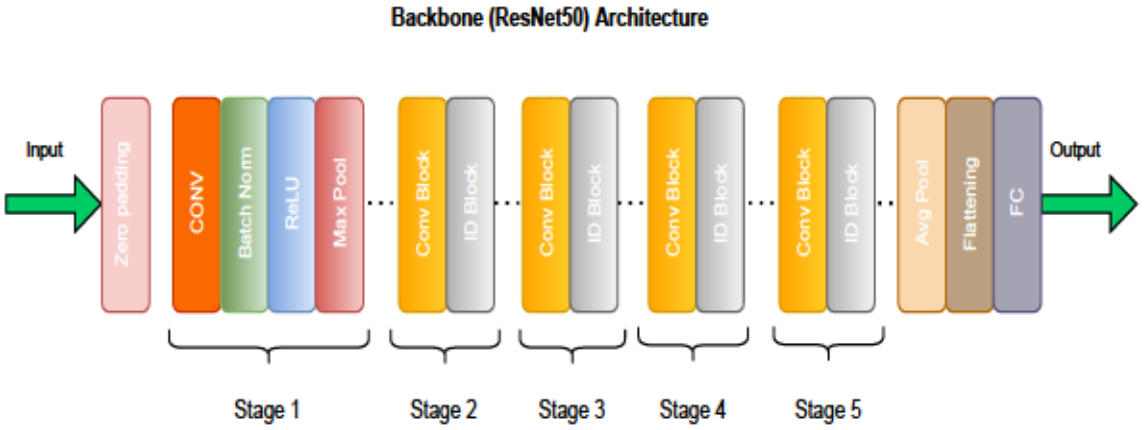

Figure 5.2: DETR backbone.

representation. This is then fed into the DETR transformer decoder, which uses object queries to transform these encoded features into 2D object detections in the form of bounding boxes, with class labels for each of the detected objects. The core role of the DETR module in this framework is, therefore, to establish object identification and its classification based on appearance from camera images.

5.3.2 3D Object Detection Framework

Our architecture for the 3D detection is adapted from the Yin et al. (2021) approach, which is meant to process raw LiDAR point cloud data in three-dimensional object detection. Before processing, voxelization pre-processing converts the point cloud into structured grid format, thereby making it amenable to processing through convolutional layers. After that, the convolutional layers aim to extract notable 3D features that are subsequently used to produce 3D bounding boxes and class labels, encapsulating the spatial features of the detected objects. The role of 3D framework is hence crucial in capturing the depth and geometric information of the scene, a pre-requisite to accurately identify objects in 3D.

5.3.3 Feature Fusion and Integration with Transformers

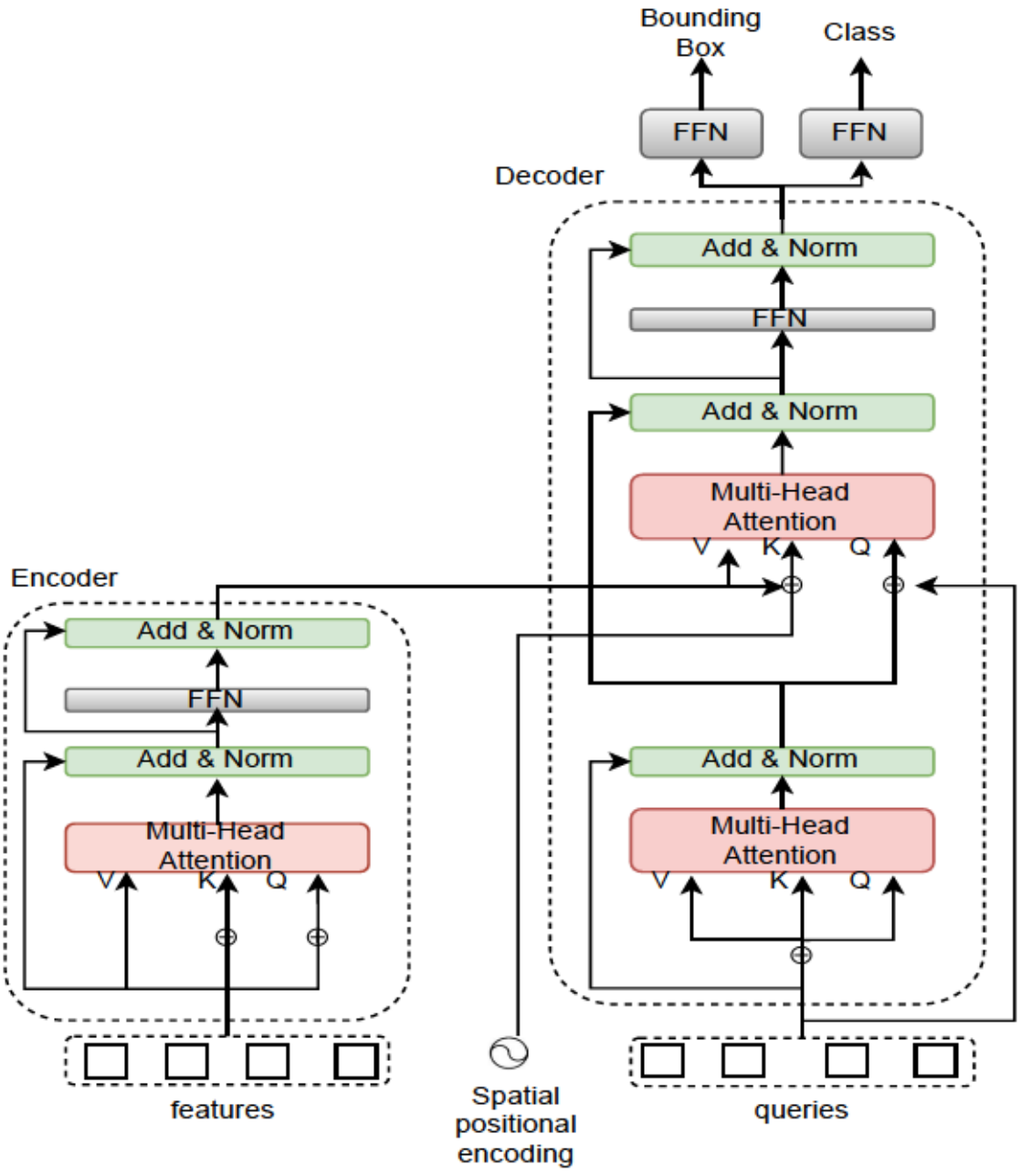

Figure 5.3: Encoder decoder architecture of the integrated transformer.

The main novelty of our approach lies in the integrated transformer module, which combines the outputs from both the DETR and 3D detection modules into unified multi-modal tracking. Whilst the DETR module solely focuses on 2D object detection, the encoder in the integrated transformer instead processes 2D and 3D features, separately and with the objective of refining and encoding these features without losing their individual characteristics. It is ensured by this approach that the distinctive information conveyed by both the 2D and 3D data is preserved and enriches the feature representation for further effective integration.

The pipeline begins with independent feature extraction from 2D images and 3D point clouds. DETR extracts spatial embeddings that emphasize appearance, shape, and texture details, while the 3D detection module, such as CenterPoint, processes point cloud data to extract spatial and geometric features. These features represent complementary aspects of the scene: the visual information from the camera and the structural information from LiDAR.

The integrated transformer combines the information from both modalities at the decoder stage. The refined features from the 2D and 3D data are combined using a cross-attention mechanism, aligning the two modalities and integrating them into a unified representation.

Cross-attention works by forming queries from one modality (e.g., 2D features) and keys and values from the other modality (e.g., 3D features). The computed attention weights highlight the relevance of specific features in one modality with respect to the other. This dynamic alignment enables the model to focus on the most relevant features from both modalities, ensuring robust tracking performance even under challenging conditions like occlusions or appearance changes.

The decoder leverages object queries to detect new objects in the scene and track queries to maintain the identities of objects detected in previous frames. By dynamically updating the importance of each modality based on the scene context,

the cross-attention mechanism enables the model to concentrate on features that are most relevant for accurate detection and tracking.

5.3.4 Data Association and Object Tracking

Data association and object tracking are thus handled coherently within the integrated transformer decoder. Object queries identify the set of newly detected objects, whereas track queries update and maintain the states of tracked objects across time. Encompassed within this are issues of object identity management, occlusion management, and re-identification of objects that disappear and then reappear. The attention mechanisms in the decoder enable it to effectively match new detections with the existing set of tracks, providing continuity and accuracy in the tracking of objects.

In a nutshell, the integrated transformer architecture of our method proposes an end-to-end multi-object tracking that effectively fuses the multi-modal data and executes the object identity consistency over time. The proposed architecture separates feature extraction and refinement in the encoder, carrying out a thorough feature fusion and tracking in the decoder to ensure robust performance even in dynamic and complex scenarios. This architecture will make full use of both 2D and 3D data to capture a rich, accurate representation of targets and improve the tracking capability for autonomous systems as a whole.

5.4 Experimental Evaluation

5.4.1 Datasets

In this work, we evaluate the performance of the proposed method using two widely recognized datasets in the field of autonomous driving: the nuScenes

dataset and the KITTI tracking dataset.

The nuScenes dataset covers one of the large-scale, multimodal datasets that have gone through heavy annotation in a number of autonomous driving tasks. It contains synchronized data from multiple sensors, such as LiDAR and cameras, making it very suitable for 3D MOT. The scenarios in this dataset vary from different weather conditions to night time scenes, thus offering a robust benchmarking for performing autonomous driving systems in the real world. Another contribution is the KITTI tracking dataset, which focuses on urban traffic scenarios and features mainly vehicles. It provides detailed 2D and 3D annotations, which makes it very suitable for tasks involving sensor fusion, where the integration of data from different modalities is crucial. The fact that the KITTI dataset puts an emphasis on urban traffic scenes allows a thorough assessment of the tracking methods in environments typical of everyday driving conditions.

5.4.2 Evaluation Metrics

We use the standard evaluation metrics of the multi-object tracking field, specifically the CLEAR-MOT, to assess the performance of our tracking system. Among all the metrics, the most important is the Multi-Object Tracking Accuracy (MOTA) metric, because it is an inclusive metric: missed detections, false positives, and identity switches are all factors combined that give a holistic view pertaining to how effective the tracking system is performing.

Apart from MOTA, we report the averaged variants of these metrics: AMOTA and AMOTP. The averaged metrics will allow having a more specific overview concerning tracking performance for various challenging scenarios and datasets, encapsulating both accuracy and precision. This is done to maintain consistency with previously established protocols for evaluation and creates room for fair com-

parisons with previous works in the area.

$$\text{AP} = \int_0^1 \text{Precision}(r)\, dr, \tag{5.2}$$

where $\text{Precision}(r)$ represents the precision of the model at a given recall r. AP measures the area under the Precision-Recall curve, providing an overall performance metric for object detection.

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t}, \tag{5.3}$$

where:

- FN_t: Number of false negatives at time t,
- FP_t: Number of false positives at time t,
- IDSW_t: Number of identity switches at time t,
- GT_t: Number of ground-truth objects at time t.

MOTA evaluates tracking accuracy by penalizing false negatives, false positives, and identity switches, normalized by the total number of ground-truth objects.

$$\text{AMOTA} = \frac{1}{n_r} \sum_{r=1}^{n_r} \text{MOTA}(r), \tag{5.4}$$

where n_r represents the total number of recall thresholds considered. AMOTA provides the average MOTA score across multiple recall thresholds, capturing the tracking performance across varying levels of recall.

$$\text{MOTP} = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}, \tag{5.5}$$

where:

- $d_{i,t}$: Distance between the predicted and ground-truth positions of object i at time t,
- c_t: Number of correctly matched objects at time t.

MOTP measures the average localisation error of tracked objects, quantifying the precision of the spatial alignment between predictions and ground-truth data.

5.4.3 Results

The results of our experiments demonstrate the effectiveness of our method in various tracking and detection tasks. Table 5.1 shows the tracking performance of the car and pedestrian classes on the KITTI validation set. Our approach yields competitive tracking performance for both object classes. Although the results are slightly worse than the top-performing method, our approach still maintains very strong tracking capabilities.

Method	Input	Car				Ped			
		sAMOTA↑	MOTA↑	MOTP↑	IDs↓	sAMOTA↑	MOTA↑	MOTP↑	IDs↓
EagerMOT Kim et al. (2021)	2D+3D	96.93	95.29	76.97	1	92.92	93.14	73.22	36
GNN3DMOT Weng, Wang, Man & Kitani (2020)	2D+3D	96.03	94.70	75.93	10	-	-	-	-
mmMOT Zhang et al. (2019)	2D+3D	93.68	84.70	74.77	12	-	-	-	-
FANTrack Baser et al. (2019)	3D	82.97	74.30	72.45	202	-	-	-	-
AB3DMOT Weng, Wang, Held & Kitani (2020)	3D	91.78	83.35	78.17	1	73.18	66.98	67.77	1
Ours	2D+3D	93.02	87.62	76.63	7	76.18	71.07	70.63	23

Table 5.1: Tracking performance comparison for car and pedestrian classes on KITTI val set

Method	AP↑	ATE (m)↓	ASE (1-IOU)↓
PointPillars Lang et al. (2019)	0.684	0.281	0.164
TransMOT Ruppel et al. (2022)	0.727	0.284	0.161
Our Method	0.736	0.268	0.152

Table 5.2: Object detection results for the car class

Table 5.2 gives the performance of object detection for the class car. It is straightforward to see from Table 5.2 that the Average Precision (AP) of our approach is the highest among the compared methods. Therefore, it outperforms others in object detection. This is because effective fusing of the multi-modal data boosts both the accuracy and robustness of car detection for our method.

Method	AMOTA↑	TP↑	FP↓	FN↓	IDS↓
CenterPoint Yin et al. (2021)	63.8	95877	18612	22928	760
EagerMOT Kim et al. (2021)	67.7	93484	17705	24925	1156
AlphaTrack Zeng et al. (2021)	69.3	95851	18421	22996	718
Ours	65.4	94454	17963	22617	1013

Table 5.3: Tracking results comparison on the Nuscenes.

Table 5.3 comparison of tracking results on the nuScenes dataset. Our approach ranks a respective second with an AMOTA score of 65.4, behind the 69.3 presented by the method in Zeng et al. (2021). Although this places us slightly behind some of the leading methods, performance shows a stable tracking capability considering the nuScenes dataset is challenging. Notice that our approach tends to have lower numbers of FP compared to EagerMOT, which underlines higher precision in object tracking.

In short, our approach performed consistently on all datasets and metrics of tracking. While the first ranking might not be possible on all metrics, it showed a very balanced strength of both detection accuracy and stability in tracking, especially with fewer identity switches and false positives. Due to this balance, our approach is robust and reliable, especially for complex scenarios with multiple object types in various environments. Figure 5.4 shows some qualitative results of tracking. Training and validation losses can be seen in Figure 5.5 and Figure 5.6, respectively.

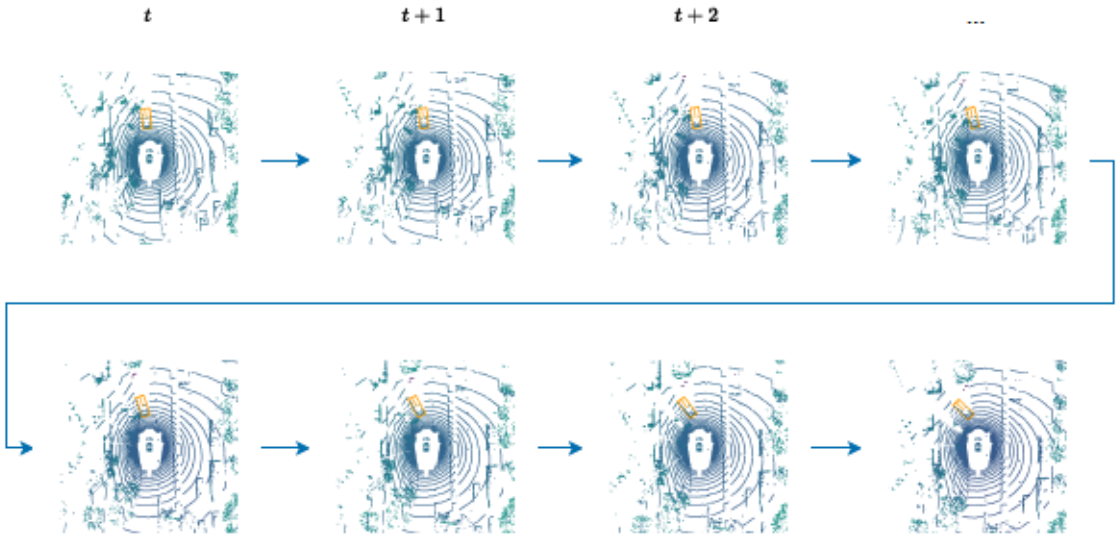

Figure 5.4: Qualitative result showing the tracking through time.

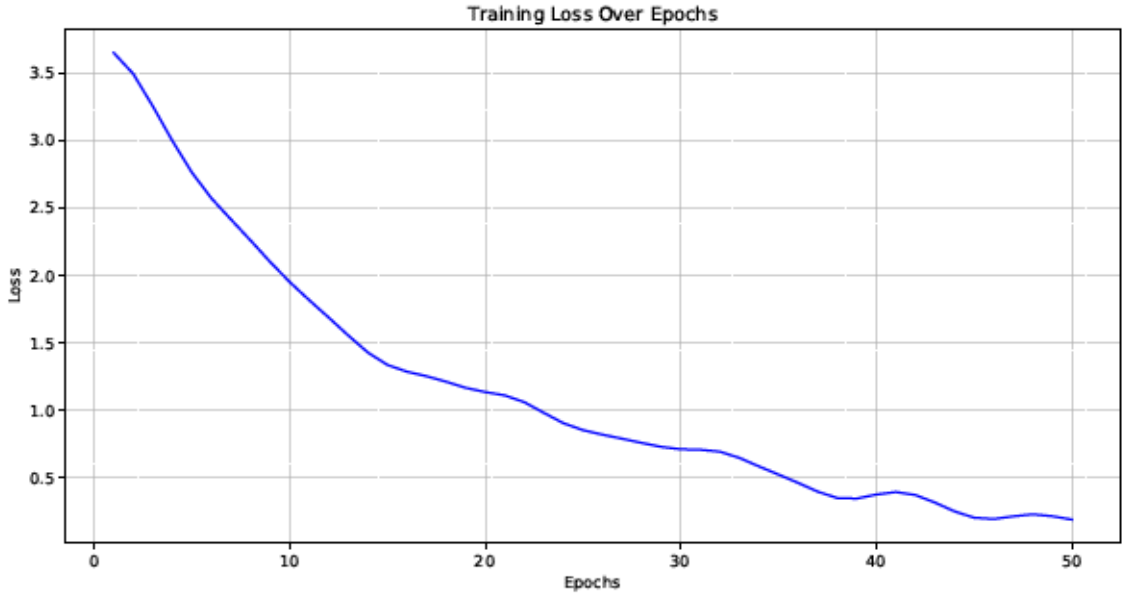

Figure 5.5: Training loss per number of epochs during the process.

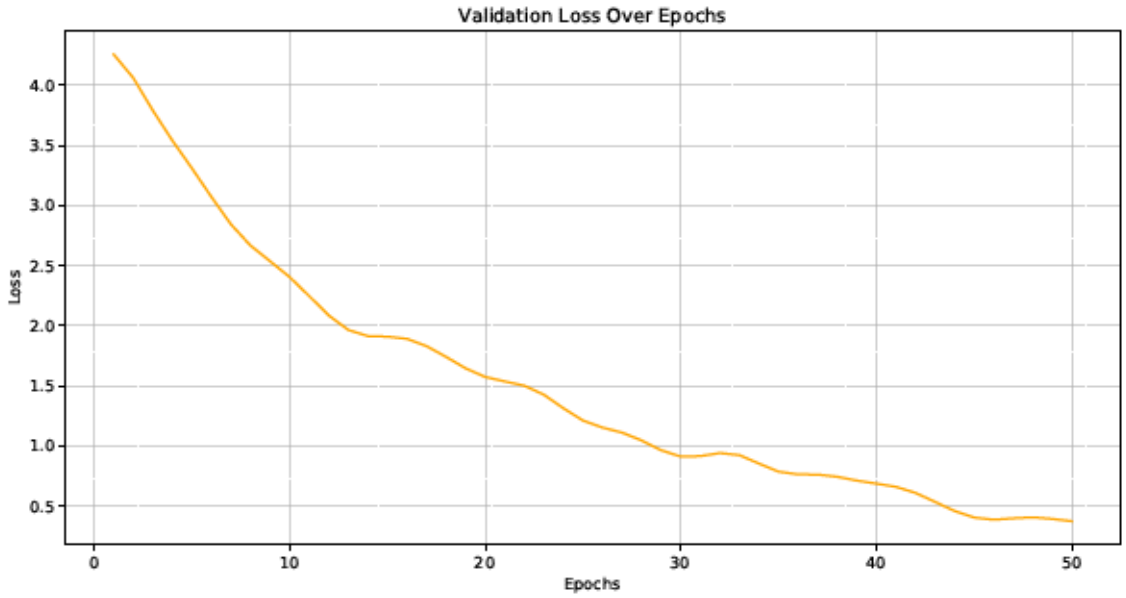

Figure 5.6: Validation loss per number of epochs during the process.

5.5 Conclusion

This chapter has introduced the improved framework for multi-object tracking, effectively combining transformer-based object detection with advanced sensor fusion. We design the transformers in a tracking-by-attention paradigm to exploit their capability of modeling global context and higher-order dependencies over both 2D and 3D data. This yields significant improvements in tracking accuracy and robustness, particularly in dynamic scenes and challenging conditions.

We evaluated this approach on two large, well-known datasets in autonomous driving: nuScenes and KITTI. These experimental results certify that our proposed method delivers competitive performance for challenging multi-object tracking problems in various dynamic scenarios. The incorporation of 2D and 3D features improves the detection accuracy and tracking consistency, while the synergy of these data modalities enables deeper insights into the environment. This results in a more reliable and effective tracking output, a critical factor for autonomous driving systems.

A notable strength of the proposed framework lies in its ability to leverage transformer architectures for capturing complex relationships and dependencies among objects. This capability is particularly crucial in autonomous driving applications, where understanding the interactions between multiple objects is essential for safe navigation. By effectively utilising self-attention mechanisms and cross-attention layers, the framework ensures that appearance and spatial data from different sensor modalities are seamlessly fused, enabling comprehensive environmental interpretation.

While the results are promising, there remains room for further improvement. optimising the network to enhance performance and increase overall accuracy for

real-time applications is an important future direction. Another avenue of research could be how to embed other sensor modalities, such as radar, in a way that allows richer environmental perception and enhances the robustness across a variety of conditions. For instance, radar sensors can provide critical information on those climatic conditions when most of the other sensors, such as cameras and LiDAR, fail to operate efficiently. Moreover, exploring the integration of emerging data sources, such as thermal cameras or GPS, could expand the scope of multi-modal data fusion, enabling more extensive environmental perception and enhancing overall system reliability and safety in unpredictable conditions.

The findings of this chapter form a pivotal bridge between the segmentation techniques discussed in the previous chapters and the motion forecasting approaches explored in subsequent work. The emphasis on integrating multi-modal data and leveraging transformers for robust tracking informed our exploration of motion prediction, where understanding object trajectories and interactions becomes paramount. The demonstrated success of attention-based fusion techniques inspired the development of novel transformer architectures for motion forecasting, aiming to model not just spatial relationships but also temporal dynamics.

In summary, the proposed framework marks a significant advancement in multi-object tracking, showcasing the effectiveness of transformer-based architectures in fusing 2D and 3D sensor data. By providing tangible improvements in tracking performance, this work lays a strong foundation for subsequent research on understanding object trajectories and interactions. The insights gained here, particularly in leveraging transformers to model complex spatial relationships, informed the development of novel transformer architectures for motion forecasting. In the next chapter, the findings and methods developed here are extended to tackle the challenge of motion forecasting. Building on the insights gained from tracking object interactions, the next chapter introduces innovative techniques such as anchored

goal queries and localised attention mechanisms to predict agent trajectories with high accuracy. These methods leverage the strengths of transformer architectures to address the complex dynamics of multi-agent interactions, contributing to safer and more reliable autonomous driving systems. These contributions align with the broader narrative of the thesis, highlighting the evolution from tracking to forecasting as part of building robust perception systems for autonomous driving.

CHAPTER 6

Motion Forecasting

This chapter provides an in-depth insight into motion forecasting, one of the key tasks in autonomous driving, that basically forecasts of future trajectories of traffic agents like vehicles, pedestrians, and cyclists. It presents the Intention Refined Motion Transformer (IRMTR), a novel architecture that combines local and global contextual information to improve accuracy in forecasts. This framework uses anchored goal queries to propose possible future positions of the agents and localised attention mechanisms that dynamically capture agent interactions with the environment. The model makes use of high-definition map information and real-time sensor data together to develop a comprehensive representation of the driving environment. It can make accurate predictions of complex multi-agent interactions such as lane changes, merges, and intersection navigation. Extensive evaluations using the Argoverse dataset result in significant improvements in both high accuracy and robustness. Ablation studies prioritize investigating key components such as anchored goal queries and localised attention, which refine predictions of IRMTR as a robust solution toward motion forecasting in real-world driving scenarios.

6.1 Motivation

While autonomous driving technology is rapidly improving, it still faces one of the big challenges: maintaining safety in a dynamic, unpredictable environment. A self-driving vehicle has to be able not only to perceive and interpret what happens around it but also to predict what the future behavior will be like of other road participants, such as vehicles, cyclists, or pedestrians. This leads to collision avoidance and informed decision-making in completing complex traffic situations. This gets very difficult because there are so many different ways one person may intend on doing something or the other. Each vehicle or pedestrian's future trajectory depends on interactions with others and adherence to the local traffic regulations, making tasks of motion forecasting much more complex.

In our previous work related to LiDAR segmentation and multi-object tracking, advanced techniques have been identified to allow deep learning models based on CNN and transformers to compete with classic ones in detecting and tracking of objects at runtime with high accuracy. This is an effort to improve the perception of the proximal environment that surrounds a vehicle through the processing of LiDAR point cloud and cameras for an accurate and high-resolution representation of the objects surrounding the vehicle. While these techniques showed good results in object detection and tracking, they did not fulfill the key requirement of forecasting future states of those objects, which forms the very basis of independent decision-making in a dynamic environment.

Building on our work, this research moves beyond segmentation and tracking into the next logical step in autonomous driving, motion forecasting.

In multi-agent scenarios, the behavior of each agent is influenced by interactions with objects in the scene and specific rules of traffic at the location. This

makes the prediction of diverse behaviors in a scene quite difficult since it involves not only an understanding of the current state of each agent but also how such states would evolve according to both individual and collective dynamics. Typical motion prediction approaches can be defined in two categories, goal-based and direct regression methods. Goal-based methods Gu et al. (2021), Zhao, Gao, Lan, Sun, Sapp, Varadarajan, Shen, Shen, Chai, Schmid et al. (2021) generate a set of possible next positions for an agent, estimate the probability of each position, and then predict complete trajectories for the most likely outcomes. It may be effective in handling uncertainty of trajectory prediction and simplifying model optimization. However, the approach heavily relies on the count of pre-defined goal candidates; too few candidates lower the accuracy and too many increase computation and memory requirements significantly.

In contrast to that, direct-regression methods do not rely on predefined goals. Instead, the encoded features of the agent are used to directly predict a set of possible trajectories Ngiam et al. (2021), Varadarajan et al. (2022). This allows modeling possible future behaviors in a more flexible way, without the constraints imposed by spatial priors. However, many of these methods converge slower, since they have to regress multiple modes of motion with the same features, and can be biased to predict the most frequent behaviors during training, since this reduces the loss for training, but hurts generalization to less common scenarios. So far, we have proposed a novel approach, which overcomes the limitations of both goal-based and direct-regression methods by incorporating strengths of both approaches while mitigating weaknesses. This gave rise to the Intention Refined Motion Transformer, an advanced architecture further improving multi-agent motion accuracy and robustness in dynamic environments. IRMTR refines the intention-based mechanism with a compact set of intention points representing the most probable future positions of agents, eliminating the need for dense candidate goal

placement and greatly enhancing efficiency of the model to meet real-world application requirements.

Our method further develops the concept of intention-driven prediction through an elaborate mechanism in order to capture and embed the intention points of different agents effectively.

The IRMTR model processes scene context through a hierarchical encoding structure that embeds both local and global features and refines predictions with the embedded intentions. With this, the model is able to produce a more accurate and contextually aware prediction about agent trajectories, with the incorporation of both immediate environmental data and broader scene information. IRMTR first extracts the scene context, which includes both historical movement from the agents and static map features. This is passed through a local encoder, which embeds the immediate environment and interactions of the agent. These embeddings are further refined by the global context aggregator, which integrates broader-scene information and intention points into the prediction process. The static intention query module constitutes a significant module of this architecture that narrows the model's focus on most probable future trajectories, hence narrowing down possible outcomes for a given scenario with increased accuracy. This information is synthesized by the decoder to create highly accurate, context-aware trajectory predictions. Our experimental results involve showing that IRMTR performs competitively on the Argoverse motion forecasting dataset, showing that our model can handle most scenarios of driving. The model produces consistent, highly accurate predictions that justify the effectiveness of our intention refinement approach. With IRMTR balancing well between computational efficiency and predictive accuracy, this presents an effective solution toward real-time applications in autonomous driving and helps to push the limits further for much safer and reliable self-driving cars.

6.2 Related Work

In the task of autonomous driving, enabling motion forecasting requires to learn rich and detailed representations for diverse elements in a traffic scene, ranging from high-definition maps to the historical trajectories of agents. Indeed, accurate predictions depend on the model being able to understand both the static spatial layout of the environment and the dynamic interactions between the moving objects.

Most previous work has taken as input to their models rasterized versions of the scene. In this paradigm, an image of the world is rendered in bird's eye view with the various map elements-lane markings, crosswalks, traffic lights-all extracted from HD maps and represented in different colors or through segmentation masks Cui et al. (2019), Djuric et al. (2020), Chai et al. (2019), Bansal et al. (2018), Salzmann et al. (2020), Hong et al. (2019). It is done by either appending the past trajectories of agents as additional channels in the images Chai et al. (2019), Cui et al. (2019), or by processing them using temporal models such as recurrent neural networks Rhinehart et al. (2019), Alahi et al. (2016), Rhinehart et al. (2018). These works then adopt commonly used image processing models like EfficientNet Tan & Le (2019), DenseNet Huang et al. (2017), ResNet He et al. (2016) and XceptionChollet (2017) to learn the representations of these rasterized scenes.

While the vision based approaches are good candidates to leverage from the rich history of methods developed in computer vision and can easily accommodate pre-trained image models, they usually tend to be more computationally expensive and typically fail to capture fine grained details and interactions. This has fostered the growth of vectorized methods, which have become popular owing to their efficiency in encoding sparse representations and capturing complicated structural

information of a scene Ye et al. (2021), Liang et al. (2020), Gao et al. (2020). The vectorized methods model the scene as a set of entities where each entity is defined by semantic and geometric attributes. They learn the relationships between entities. For instance, LaneGCN Liang et al. (2020) constructs a lane graph to capture the topology of a map effectively. VectorNet Gao et al. (2020) models both road maps and agent trajectories as polylines. Most recent literature follows this approach Varadarajan et al. (2022), Ngiam et al. (2021), Sun et al. (2022), Gu et al. (2021) since it is efficient in terms of computational cost and scalability.

Our scene representation follows the vectorized approach, but it involves a special method proposed in HiVT Zhou et al. (2022), where all vectorized entities are relatively defined with respect to their positions. This translation-invariant representation allows the model to generalize much better across different scenes and overcome many previous pruning works in terms of robustness.

Besides the representation of future scenes, there are various ways in which researchers have modeled multimodal future motions. Early works in this field Tang & Salakhutdinov (2019), Rhinehart et al. (2019, 2018), Alahi et al. (2016), Gupta et al. (2018) concentrated on producing sets of trajectory samples to approximate the output distribution. Accordingly, this sampling-based approach aims to capture a wide range of possible futures by generating diverse trajectory hypotheses. Other approaches utilise Gaussian Mixture Models to parameterize multimodal predictions; this typically yields a more compact representation of the distribution Phan-Minh et al. (2020), Salzmann et al. (2020), Mercat et al. (2020), Chai et al. (2019), Hong et al. (2019). That allows for effective modeling of uncertainties and invariances of agent behaviors. The HOME series Gilles et al. (2021, 2022) generates trajectories by sampling from a predicted heatmap, while IntentNet Casas et al. (2018) frames intention prediction as a classification task involving high-level actions. Recent developments have introduced regional training

methodologies Liu et al. (2021), goal-based approaches such as Mangalam et al. (2020), Fang et al. (2020), Rhinehart et al. (2019), Zhao, Gao, Lan, Sun, Sapp, Varadarajan, Shen, Shen, Chai, Schmid et al. (2021) are first estimating a number of possible goal points for the agents and then full trajectory proposals for each of these goals.

With the flexibility of transformers in modeling long-range dependencies and relations, they have seen extensive applications in natural language processing Devlin et al. (2018), Bao et al. (2021), Brown (2020), and more recently in computer vision Zeng et al. (2022), Wang, Xu, Narasimhan & Wang (2021), Carion et al. (2020), Wang et al. (2018), Dosovitskiy et al. (2020). Due to their success in these domains, transformers have been retrospectively adapted for motion prediction, modeling with great success spatial relationships, temporal dependencies, and interactions between agents and map elements Yu et al. (2020), Yuan et al. (2021), Ngiam et al. (2021), Mercat et al. (2020), Liu et al. (2021), Li et al. (2020), Giuliari et al. (2021).

Our proposed architecture, the Intention-Refined Motion Transformer, extends existing transformer-based models with a novel local and global context refinement process integrated with an intention-driven forecasting solution. In contrast to previous approaches that focus either on spatial or temporal dependencies, our contribution is represented along a hierarchical structure that combines both. IRMTR refines motion predictions by anchored goal queries that allows it to effectively obtain multi-scale interactions among agents and their environments. This ensures a very substantial improvement in prediction accuracy while maintaining computational efficiency adequate for application in real-time settings.

By incorporating the intention points directly into the prediction process, IRMTR enhances the model with more capability to reason about possible future positions of agents in a socially aware way, which is especially important in complex traf-

fic scenarios where interpreting multiple agents' intentions and possible actions play a vital role in safe navigation. The IRMTR's design demonstrates competitive performance across diverse traffic situations, underscoring the robustness and effectiveness of our streamlined and efficient architecture. This represents a significant further step toward improved motion forecasting for building safer and more reliable autonomous driving systems.

6.3 Intention Refined Motion Transformer (IRMTR)

IRMTR introduces a novel approach to motion through a combination of innovative components. A defining feature of this architecture is the use of anchored goal queries, a compact set of probable future positions for traffic agents that significantly enhance prediction efficiency and accuracy. Unlike other methods that rely on dense candidate goal placements, IRMTR leverages a GMM clustering algorithm to generate intention points. This choice ensures that the clustered intention points are flexible and probabilistically informed, effectively representing diverse trajectory endpoints while maintaining computational efficiency. By adopting GMM clustering, the architecture captures the inherent uncertainty and overlapping nature of agent trajectories in real-world scenarios, setting it apart from rigid clustering methods like K-means.

Furthermore, the architecture employs a localised attention mechanism that dynamically prioritizes immediate interactions between agents and their environment, improving fine-grained understanding while maintaining computational efficiency. Additionally, IRMTR introduces a hierarchical encoding structure to process local and global contexts separately, enabling effective handling of both detailed agent interactions and broader scene dynamics. This multi-faceted approach allows the model to reason about socially aware interactions and predict trajec-

tories accurately even in complex scenarios like intersections, lane changes, and merges.

6.3.1 Overall Framework

An overview of our IRMTR framework is illustrated in Figure 6.1. In this framework, we propose a systematic structuring of the input that incorporates both agents' previous trajectories and relevant map features; this allows our model to handle challenging motion forecasting in dynamic traffic effectively. With the proposed representation, our framework can capture and process multi-scale spatial-temporal information in multiple steps and achieve high accuracy in trajectory predictions.

First, the local encoder processes the data from each agent to compute its local context feature. All these include information about the ego-motion, interaction with nearby agents, and the structure of the road, capturing the immediate surroundings around every agent. A global contextual aggregator then aggregates these local features, enriching the representation at each agent to capture long-range interactions and entire-scene dynamics. Also, this aggregator has an important role in understanding the context of the scene in a broader aspect, important for accurate prediction of motions.

Simultaneously, an anchored goal query module refines the information by incorporating predefined intention points, which guide the model towards more precise trajectory predictions. By incorporating these intention points, the model narrows its focus to the most probable paths and hence reduces uncertainty, thereby making accurate predictions. Finally, enhanced representations are decoded to obtain multi-modal trajectory predictions of all agents, effectively modeling the inherent uncertainty in dynamic traffic scenarios.

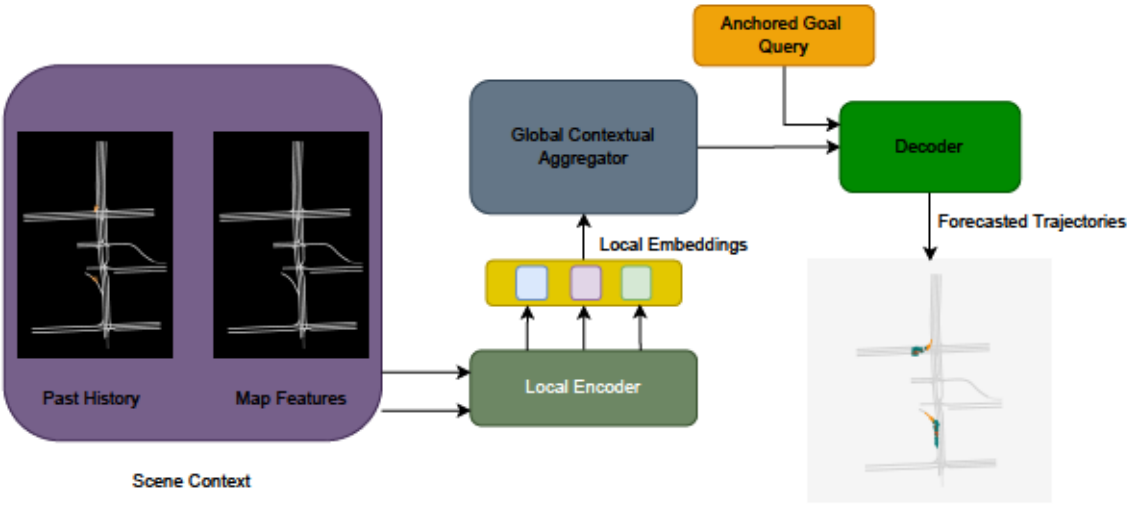

Figure 6.1: Our model architecture. Different encoder modules used for different tasks and then the features are combined using a cross attention module.

6.3.2 Environmental Context Encoding

In the IRMTR framework, we represent a traffic scene by encoding both the spatial and temporal characteristics of agents and map features. To achieve this, the scene is decomposed into vectorized entities, including the trajectory segments of moving agents and the lane segments that make up the road network. Unlike traditional methods that rely on absolute positions, our approach uses a relational representation that captures the geometric properties of these entities, ensuring robustness to environmental transformations such as translations.

For each agent a, its trajectory over time is represented by a sequence of relative displacements $\{\mathbf{u}_a^{(t)}\}_{t=1}^{T}$, where $\mathbf{u}_a^{(t)} = \mathbf{c}_a^{(t)} - \mathbf{c}_a^{(t-1)}$ and $\mathbf{c}_a^{(t)} \in \mathbb{R}^2$ denotes the agent's position at time step t. Here, T represents the total number of historical steps considered. Each lane segment λ is represented by its geometric attribute as

$$\mathbf{h}_\lambda = \mathbf{q}_\lambda^e - \mathbf{q}_\lambda^s, \tag{6.1}$$

where $\mathbf{q}_\lambda^s$ and $\mathbf{q}_\lambda^e \in \mathbb{R}^2$ are the starting and ending points of the lane segment in Equation 6.1, respectively. This relational encoding inherently preserves translation invariance, which is crucial for consistent scene interpretation across varying spatial contexts.

By arranging the scene in this manner, our representation effectively captures the relative spatial configuring between entities, such as how an agent's trajectory relates to its surrounding environment. This ensures that the learned representations remain robust, so the model can generalize across different driving scenarios and is insensitive to absolute positional shifts. It guarantees that any subsequent processing step respects inherent geometric properties of the scene for more reliable motion prediction.

6.3.3 Local Encoder for Scene Context and Agent Interaction

The local encoder in our model is fundamental for encoding the spatial structure of the environment and capturing both agent-agent and agent-environment interactions. Rather than relying on predefined local regions, we utilise a localised attention mechanism that dynamically identifies and focuses on the most pertinent interactions within the scene. This approach enables the model to be both memory-efficient and highly accurate by concentrating computational resources on interactions most likely to influence the agent's behavior.

Agent-Environment Interaction

For each agent a at time step t, the interaction with its environment is represented by its trajectory segment $\mathbf{u}_a^t - \mathbf{u}_a^{t-1}$, which serves as a directional reference vector. To ensure consistency in our model's predictions, regardless of the agent's orientation, we apply a rotation matrix $\mathbf{R}_a$ based on the orientation θ_a of this reference

vector. This rotation matrix is defined as:

$$\mathbf{R}_a = \begin{pmatrix} \cos(\theta_a) & -\sin(\theta_a) \\ \sin(\theta_a) & \cos(\theta_a) \end{pmatrix} \tag{6.2}$$

This matrix is used to align the trajectory data with a consistent reference frame, ensuring that the agent's movements are interpreted accurately in relation to the rest of the environment. The trajectory features are then encoded using multi-Layer perceptrons, where the central agent's features are processed by the MLP ϕ_{local}, and the features of the neighboring agents, adjusted relative to the central agent, are encoded by ϕ_{neighbor}.

The encoded features $\mathbf{h}_a^t$ for the central agent at time t and $\mathbf{h}_j^t$ for a neighboring agent j are given by:

$$\mathbf{h}_a^t = \phi_{\text{local}}\left(\mathbf{R}_a^T\left(\mathbf{u}_a^t - \mathbf{u}_a^{t-1}\right), \mathbf{x}_a\right) \tag{6.3}$$

$$\mathbf{h}_j^t = \phi_{\text{neighbor}}\left(\left[\mathbf{R}_a^T\left(\mathbf{u}_j^t - \mathbf{u}_j^{t-1}\right), \mathbf{R}_a^T(\mathbf{u}_j^t - \mathbf{u}_a^t), \mathbf{x}_j\right]\right) \tag{6.4}$$

where $\mathbf{x}_a$ and $\mathbf{x}_j$ are the semantic attributes of agents a and j, and $\mathbf{e}_{aj}$ is the edge attribute representing the spatial relation between these agents.

Local Scene Context Encoding with localised Attention

The localised attention mechanism employed in our model introduces significant adaptations tailored specifically to the challenges of multi-agent motion forecasting in autonomous driving. Our approach integrates this mechanism into a hierarchical encoder-decoder framework that captures both immediate and long-range interactions. By dynamically focusing on the most relevant neighbors and spatial features, our implementation ensures that critical agent-environment and agent-agent interactions are captured effectively. This method avoids the inefficiencies

of global attention by concentrating on the interactions that are most likely to influence the agent's future trajectory. This integration, coupled with the broader architecture—including the temporal encoder, global contextual aggregator, and anchored goal queries—results in a system suited for real-time, context-aware trajectory prediction in complex traffic scenarios.

The encoded features from the central agent and its neighbors are used to compute the query, key, and value vectors required for the scaled dot-product attention mechanism. Specifically:

$$\mathbf{q}_a^t = \mathbf{W}^Q \mathbf{h}_a^t, \tag{6.5}$$

$$\mathbf{k}_{aj}^t = \mathbf{W}^K \mathbf{h}_{aj}^t, \tag{6.6}$$

$$\mathbf{v}_{aj}^t = \mathbf{W}^V \mathbf{h}_{aj}^t \tag{6.7}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are learnable matrices for linear projections, and $\mathbf{q}_a^t$, $\mathbf{k}_{aj}^t$, and $\mathbf{v}_{aj}^t$ are the query, key, and value vectors, respectively.

The attention score α_a^t between the central agent a and its neighboring agent j is computed using:

$$\alpha_a^t = \text{softmax}\left(\frac{\mathbf{q}_a^t \cdot \mathbf{k}_{aj}^t}{\sqrt{d_k}}\right) \tag{6.8}$$

where d_k is the dimensionality of the key vectors.

The updated embedding for the central agent $\mathbf{g}_a^t$ is computed by combining its own embedding with the weighted sum of the value vectors from its neighbors:

$$\mathbf{g}_a^t = \mathbf{h}_a^t + \sum_{j \in \mathcal{N}_a} \alpha_{aj}^t \mathbf{v}_{aj}^t \tag{6.9}$$

This localised attention mechanism allows the model to efficiently capture critical spatial interactions, thereby improving the accuracy of its motion predictions.

The encoded local context is then utilised in subsequent layers to further refine the agent's trajectory predictions, ensuring that the model effectively leverages both immediate and broader environmental cues.

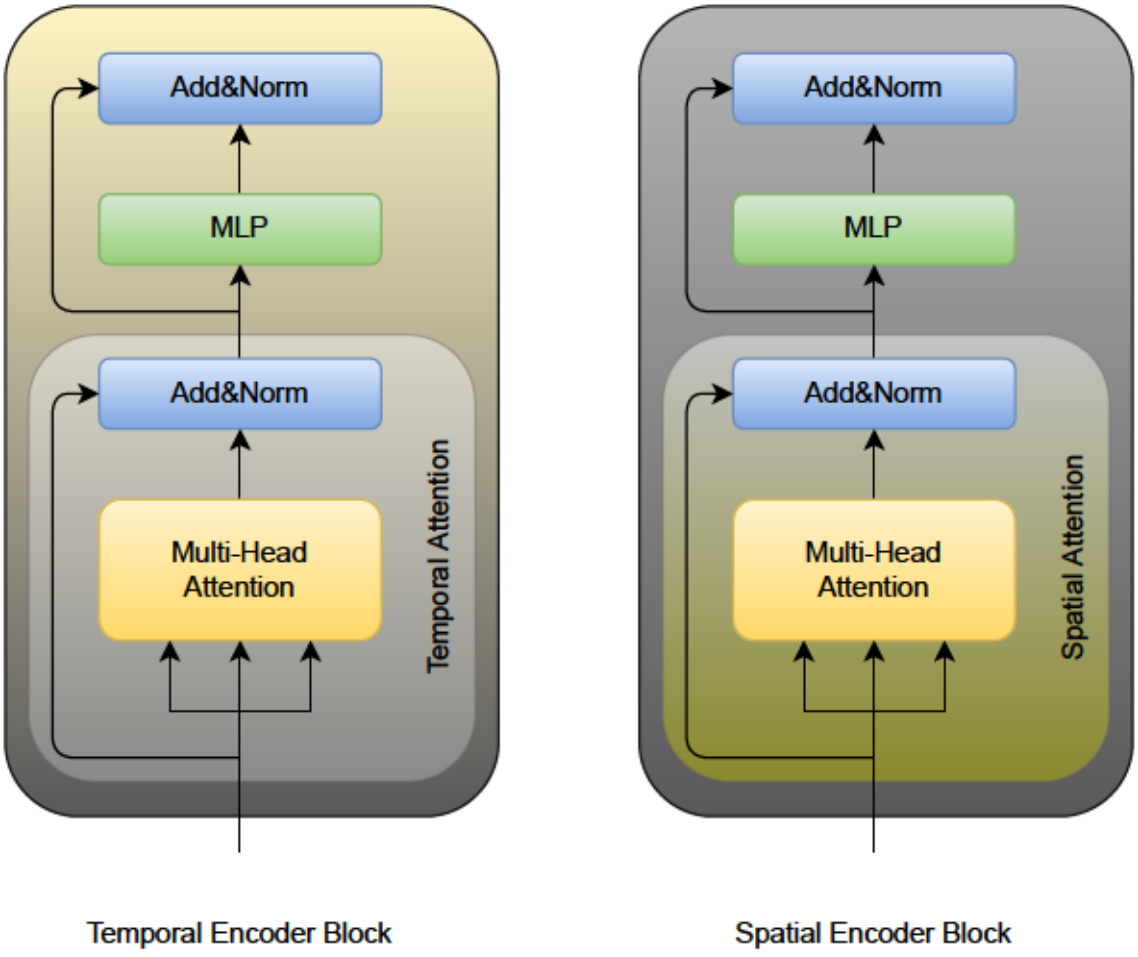

Figure 6.2: Encoder blocks inside the local encoder.

Temporal Encoder Block

To effectively capture the temporal dynamics inherent in motion prediction, we utilise a Temporal Transformer Encoder that processes the sequence of embeddings derived from the spatial context encoder across different time steps. For each central agent a, the input sequence to this module is composed of embed-

dings $\mathbf{u}_a^t$ at each time step t, where $t = 1, \ldots, T$.

In our framework, similar to the BERT architecture Devlin et al. (2018), we append a learnable token $\mathbf{u}_{T+1}^a$ to the end of the input sequence. This token acts as a global representation summarizing the temporal information of the agent's trajectory. We further enrich these embeddings by adding positional encodings to account for the sequential nature of the input data.

The resulting sequence $\mathbf{S}^a \in \mathbb{R}^{(T+1)\times d_k}$ is then passed through the Temporal Attention block, which operates as follows:

$$\mathbf{Q}_a = \mathbf{S}^a \mathbf{W}^{Q_{\text{time}}}, \tag{6.10}$$

$$\mathbf{K}_a = \mathbf{S}^a \mathbf{W}^{K_{\text{time}}}, \tag{6.11}$$

$$\mathbf{V}_a = \mathbf{S}^a \mathbf{W}^{V_{\text{time}}}, \tag{6.12}$$

$$\hat{\mathbf{S}}_a = \text{softmax}\left(\frac{\mathbf{Q}_a \mathbf{K}_a^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}_a, \tag{6.13}$$

where $\mathbf{W}^{Q_{\text{time}}}, \mathbf{W}^{K_{\text{time}}}, \mathbf{W}^{V_{\text{time}}} \in \mathbb{R}^{d_k \times d_k}$ are learnable matrices for the query, key, and value projections, and $\mathbf{M} \in \mathbb{R}^{(T+1)\times(T+1)}$ is a temporal mask that restricts the tokens from attending to future time steps:

$$\mathbf{M}_{uv} = \begin{cases} -\infty & \text{if } u < v, \\ 0 & \text{otherwise.} \end{cases} \tag{6.14}$$

This mechanism ensures that the attention mechanism only considers past and present information when predicting the future trajectory, preserving the causal structure of the sequence. The output of this temporal attention block, which encapsulates the spatial-temporal features of the scene, is subsequently fed into the next stages of our model for trajectory forecasting.

6.3.4 Global Contextual Aggregator

To capture long-range dependencies and global interactions among agents in the scene, we introduce a global contextual aggregator. Our approach aggregates global information through a holistic summarization of agent embeddings using a pooling operation, allowing for comprehensive context integration.

We generate a global context vector that encapsulates the collective dynamics of all agents by applying a pooling operation:

$$\mathbf{c}_{\text{global}} = \text{Pooling}\left(\{h_a\}_{a=1}^{N}\right) \tag{6.15}$$

where:

- h_a: Local embedding of agent a from previous encoders.
- N: Total number of agents in the scene.
- Pooling$(\cdot)$: A permutation-invariant aggregation function.

In our model, we utilise mean pooling as the pooling operation due to its simplicity and effectiveness:

$$\mathbf{c}_{\text{global}} = \frac{1}{N}\sum_{a=1}^{N} h_a \tag{6.16}$$

This mean pooling operation ensures that the global context vector is a fair and unbiased summary of all agent embeddings, capturing the overall scene context without being influenced by the order of agents or dominated by any particular agent.

Each agent's embedding is then refined by integrating it with the global context vector:

$$\tilde{h}_a = \text{MLP}_{\text{refine}}\left([h_a; \mathbf{c}_{\text{global}}]\right) \tag{6.17}$$

where:

- $[h_a; \mathbf{c}_{\text{global}}]$: Concatenation of the agent's local embedding h_a and the global context vector $\mathbf{c}_{\text{global}}$.
- $\text{MLP}_{\text{refine}}$: A multi-layer perceptron that learns to fuse local and global information.
- $\tilde{h}_a$: The refined embedding for agent a.

By first defining a general pooling operation and then specifying that we use mean pooling, we clarify our methodology and the rationale behind our choice. The global context vector $\mathbf{c}_{\text{global}}$ represents the collective behavior of the scene, capturing average movements, interactions, and overall dynamics.

Integrating this global context with each agent's local embedding allows the model to enhance the agent's representation with information about the broader scene. This is particularly beneficial in scenarios where the collective behavior influences individual agents, such as in traffic flow or crowd movements.

This method is computationally efficient and avoids the complexity of computing pairwise interactions between agents, making it suitable for real-time applications and large-scale scenes.

6.3.5 Decoder with Anchored Goal Queries

Our proposed model utilises a decoder architecture specifically designed to refine motion predictions by incorporating anchored goal queries. These queries help focus the attention mechanism on relevant future positions, improving the accuracy of trajectory predictions.

Problem Formulation and Notation

In the context of motion prediction for autonomous driving, we aim to predict the future trajectories of agents over a fixed prediction horizon. We define:

- N: Number of agents in the scene.
- T: Length of the observed trajectory (number of historical time steps).
- H: Prediction horizon (number of future time steps).
- K: Number of modes or anchored goal queries considered for multimodal prediction.
- $\mathbf{p}_{a,t} \in \mathbb{R}^2$: Position of agent a at time step t.

Time steps t range from 1 to T for observed data and from $T+1$ to $T+H$ for future predictions.

Anchored Goal Queries

To effectively localise potential motion intentions of agents, our model incorporates anchored goal queries. These queries are designed to reduce the uncertainty in predicting future trajectories by focusing on representative intention points that encapsulate possible future positions of the agents.

We generate K representative intention points $A \in \mathbb{R}^{K\times 2}$ by applying a Gaussian Mixture Model clustering algorithm on the endpoints of the ground-truth trajectories. Each anchored goal query corresponds to an implicit motion mode, capturing both the direction and velocity of the agent's possible future trajectory.

Each anchored goal query is modeled as a learnable positional embedding of the intention point, formulated as:

$$Q_A = \text{MLP}(\text{PE}(A)) \tag{6.18}$$

where:

- $\text{PE}(\cdot)$ is the sinusoidal positional encoding function.
- $Q_A \in \mathbb{R}^D$ is the learnable embedding of the anchored goal query.

Collectively, $Q_A \in \mathbb{R}^{K \times D}$ contains all goal query embeddings.

Integration with Global Contextual Aggregator

The decoder in our architecture is designed to translate the aggregated context features, which include both the global scene context and refined intention queries, into accurate future trajectory predictions for each agent. The decoder leverages the input from the global contextual aggregator and the anchored goal queries to generate multimodal trajectories.

For each agent a and anchored goal query $Q_{A,k}$, the input to the decoder is represented as:

$$C_{\text{input}}^{(a,k)} = \tilde{h}_a + Q_{A,k} \tag{6.19}$$

where:

- $\tilde{h}_a$ is the refined embedding of agent a obtained from the global contextual aggregator, calculated as:

  $$\tilde{h}_a = \text{MLP}_{\text{refine}}\left([h_a; \mathbf{c}_{\text{global}}]\right)$$

  with h_a being the local embedding of agent a, and $\mathbf{c}_{\text{global}}$ the global context vector.

- $Q_{A,k}$ is the embedding of the k-th anchored goal query.

Attention Mechanism

We employ an attention mechanism to focus on critical features from the encoder outputs. For each agent a and anchored goal query $Q_{A,k}$, we define:

Query:

$$Q_{a,k} = C_{\text{input}}^{(a,k)} W^Q \tag{6.20}$$

Key and Value:

$$K = H_{\text{enc}} W^K, \quad V = H_{\text{enc}} W^V \tag{6.21}$$

where:

- H_{enc} is the set of encoded features from the encoder or global contextual aggregator.
- W^Q, W^K, and W^V are learnable weight matrices.

The attention scores for agent a and anchored goal query $Q_{A,k}$ are computed as:

$$\alpha_{a,k} = \text{softmax}\left(\frac{Q_{A,k} K^\top}{\sqrt{d_k}}\right) \tag{6.22}$$

where d_k is the dimensionality of the key vectors.

The context vector is then obtained by:

$$\mathbf{z}_{a,k} = \alpha_{a,k} V \tag{6.23}$$

Multimodal Trajectory Prediction with GMM

For each decoder layer, we attach a prediction head to $\mathbf{z}_{a,k}$ for predicting future trajectories. As the behaviors of agents are highly multimodal, for each future

time step, we model the distribution of predicted trajectories with a GMM. Concretely, for each future time step $t \in \{1, \cdots, H\}$, we predict the parameters of each Gaussian component as follows:

$$Z_{a,k,t} = \text{MLP}(\mathbf{z}_{a,k}) \tag{6.24}$$

where $Z_{a,k,t} \in \mathbb{R}^6$ includes the parameters $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho, \pi_{a,k})$ of the bivariate Gaussian distribution $\mathcal{N}(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ for agent a and mode k at time step t.

The predicted distribution of the agent's position at time step t can be formulated as:

$$P_{a,t}(\mathbf{o}) = \sum_{k=1}^{K} \pi_{a,k} \cdot \mathcal{N}\left(\mathbf{o} \mid \mu_{a,k,t}, \Sigma_{a,k,t}\right) \tag{6.25}$$

where:

- $\mathbf{o} \in \mathbb{R}^2$ is a spatial position.
- $\mu_{a,k,t} = (\mu_x, \mu_y)$ is the mean position for agent a and mode k at time step t.
- $\Sigma_{a,k,t}$ is the covariance matrix constructed from σ_x, σ_y, ρ.
- $\pi_{a,k}$ is the mixture weight (predicted probability) for component k for agent a, satisfying $\sum_{k=1}^{K} \pi_{a,k} = 1$.

The predicted trajectories $T_{a,k}$ can be generated by extracting the predicted means $\mu_{a,k,t}$ over the future time steps.

Decoder Output

Finally, the decoder produces the predicted trajectories by applying a series of multi-layer perceptrons on the refined feature set. The output of the decoder represents the future positions of agents, considering multiple possible motion

modes. This approach enables the model to generate diverse and plausible future trajectories, contributing to safer and more reliable autonomous driving systems.

6.3.6 Training Losses

Our model is trained end-to-end using a combination of regression and classification losses to optimize the multimodal trajectory predictions. The training process is designed to minimize the prediction error while ensuring diverse trajectory hypotheses through the following components:

Regression Loss

To measure the accuracy of predicted trajectories against the ground truth, we adopt a negative log-likelihood (NLL) regression loss. For each agent a, we compute the regression loss over the Gaussian components:

$$\mathcal{L}_{\text{reg}}^{(a)} = -\frac{1}{H} \sum_{t=1}^{H} \log \left(\sum_{k=1}^{K} \pi_{a,k} \cdot \mathcal{N} \left(\mathbf{p}_{a,t} \mid \mu_{a,k,t}, \Sigma_{a,k,t} \right) \right) \tag{6.26}$$

where:

- $\mathbf{p}_{a,t}$ is the ground truth position of agent a at future time step t.
- $\pi_{a,k}$ is the mixture weight for component k for agent a.
- $\mu_{a,k,t}$ and $\Sigma_{a,k,t}$ are the mean and covariance of the Gaussian component k for agent a at time step t.

Classification Loss

The model also predicts the likelihood of each trajectory mode. To optimize these predictions, we use a cross-entropy classification loss $\mathcal{L}_{\text{cls}}$, which helps in learning the mixture weights $\pi_{a,k}$:

$$\mathcal{L}_{\text{cls}}^{(a)} = -\sum_{k=1}^{K} y_{a,k} \log \pi_{a,k} \tag{6.27}$$

where:

- $y_{a,k}$ is a binary indicator (0 or 1) that assigns the ground truth trajectory to one of the predicted modes for agent a.
- $\pi_{a,k}$ satisfies $\sum_{k=1}^{K} \pi_{a,k} = 1$ and represents the predicted probability of mode k for agent a.

Combined Loss Function

The final loss function is a weighted combination of the regression and classification losses:

$$\mathcal{L} = \frac{1}{N} \sum_{a=1}^{N} \left(\lambda_{\text{reg}} \mathcal{L}_{\text{reg}}^{(a)} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}}^{(a)} \right) \tag{6.28}$$

where:

- N is the number of agents.
- λ_{reg} and λ_{cls} are weighting factors that balance the contributions of the regression and classification losses.

This loss function ensures that the model is both accurate in its trajectory predictions and robust in distinguishing between multiple plausible futures.

Rationale Behind the Training Loss Function

The design of the training loss function in IRMTR is central to its ability to balance accuracy, multimodality, and robustness. The model employs a negative log-likelihood (NLL) regression loss, which allows it to predict a probability distribution over multiple plausible trajectories rather than a single deterministic outcome.

This is particularly important in motion forecasting, where agents often have several valid paths to choose from. The NLL loss ensures that the model captures this inherent uncertainty by modeling each trajectory as a bivariate Gaussian distribution, reflecting both spatial accuracy and multimodal behavior.

In addition to regression, the model incorporates a cross-entropy classification loss to optimize the likelihood of each predicted trajectory mode. This ensures that the model accurately prioritizes plausible modes and avoids mode collapse, where it might otherwise predict only the most frequent behaviors. By combining regression and classification losses with carefully tuned weighting factors, the training process encourages diverse trajectory predictions while maintaining precision for the most likely outcomes. This integrated loss design also aligns closely with evaluation metrics such as minADE and minFDE, ensuring that improvements in the loss function directly translate to better real-world performance.

6.4 Evaluation Outcomes

6.4.1 Experimental Setup

Intention Points Creation

In this section, we compare the two clustering techniques, k-means clustering and Gaussian Mixture Model clustering, for the purpose of generating intention points based on the AGENT class. Both methods are applied to the same dataset of endpoints to form clusters that represent the likely future paths of agents. Below are detailed observations from the visualizations, followed by a comparison of the two techniques and an explanation of why GMM was chosen.

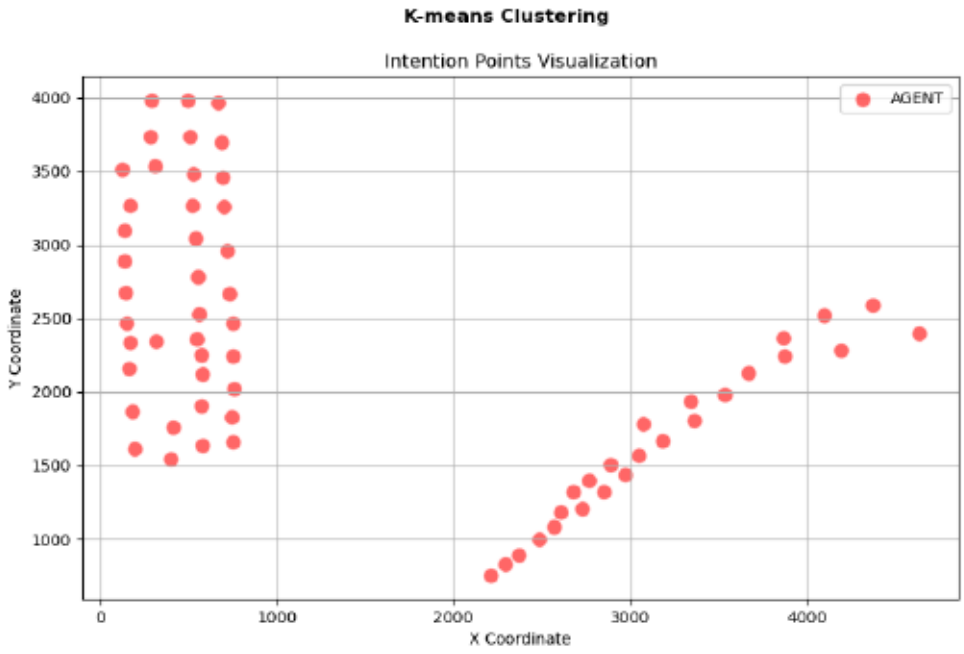

Figure 6.3: Intention points created using K-means clustering algorithm.

The clusters generated by k-means (see Figure 6.3) are more uniform, with evenly distributed intention points within the clusters. This method inherently assumes that the clusters have a spherical shape and are of similar size. As a result, the points are grouped into relatively homogeneous clusters, with clearly defined boundaries between them. The rigid assignment of each point to a single cluster is reflected in the clean, distinct clusters visible in the plot.

In contrast, GMM (see Figure 6.4) clustering provides more flexibility in the shape of the clusters. The algorithm assumes that data is generated from a mixture of Gaussian distributions, allowing clusters to take on elliptical or irregular shapes, rather than strictly spherical forms. GMM also allows for soft clustering, where each point is assigned to multiple clusters with a probability. This probabilistic assignment makes GMM more adaptable to complex or overlapping data, which is particularly evident when observing the more natural and flexible distribution of

intention points in the plot.

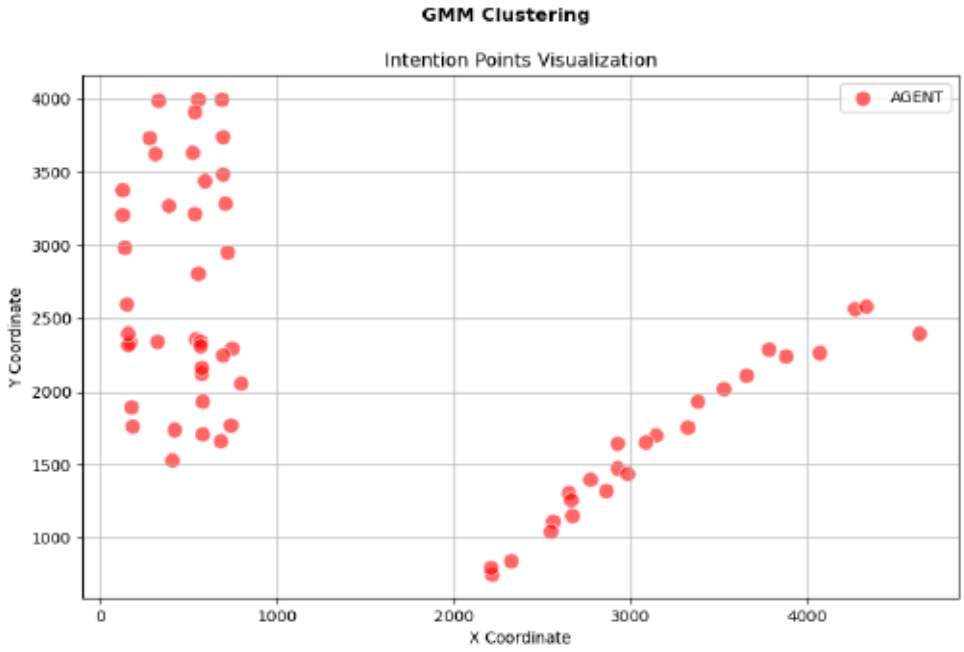

Figure 6.4: Intention points created using GMM clustering algorithm.

The k-means assumes that clusters are spherical and of equal size. This works well when the data fits these assumptions. However, real-world data, such as agent trajectories, often do not form such perfect shapes.

GMM, on the other hand, allows clusters to take on more flexible shapes. This is important in motion forecasting, where the paths that agents might take are not always uniformly distributed or confined to clear, separate regions.

In k-means, each point is strictly assigned to the nearest cluster center based on Euclidean distance. While this provides clear-cut boundaries, it can lead to inaccuracies when data points lie near the boundary between two clusters, as they are forced into a single cluster regardless of how close they might be to the others.

GMM offers a soft assignment, where each point is assigned to clusters based on a probability distribution. This probabilistic model is better suited for motion

prediction, where agents may have overlapping or ambiguous trajectories. GMM accounts for this uncertainty by assigning probabilities, rather than enforcing hard boundaries.

In k-means, overlapping data points are assigned strictly to one cluster, which can distort the cluster boundaries and result in less accurate intention points, especially when agents are traveling in close proximity or have overlapping trajectories.

GMM, with its ability to model data as a mixture of Gaussians, is naturally better at handling overlapping data points. This makes it a more suitable choice when forecasting motion in environments where agent paths frequently intersect or run close to one another, which is often the case in autonomous driving scenarios.

The k-means is computationally efficient and generally faster than GMM. Its simplicity makes it a popular choice when working with large datasets and when real-time performance is a critical factor.

GMM is computationally more expensive due to the probabilistic nature of the algorithm, which requires estimating the parameters of multiple Gaussian distributions. However, given that the intention points are computed offline (prior to real-time use), this additional computational cost is manageable and justified by the improved accuracy and flexibility in clustering.

While k-means clustering provides a fast and straightforward solution for clustering intention points, its limitations in handling non-spherical clusters, overlapping data, and its rigid assignment of points make it less suitable for complex datasets like motion forecasting. GMM offers a more flexible, accurate, and probabilistically informed approach, making it the preferred choice for generating intention points in our work. Its ability to model uncertainty and overlapping trajectories provides a significant advantage in predicting agent behavior in real-world autonomous driving scenarios.

Dataset

We evaluate the performance of our Intention Refined Motion Transformer (IRMTR) using the comprehensive Argoverse motion forecasting dataset Chang et al. (2019). This dataset is specifically designed to support research in autonomous driving, providing extensive annotations for a wide range of real-world driving scenarios. The dataset consists of 323,557 driving sequences, which include detailed trajectories of various agents, such as vehicles, cyclists, and pedestrians, alongside high-definition map data that captures the structural layout of the environment.

The dataset is split into training, validation, and test sets, with 205,942 samples in the training set, 39,472 samples in the validation set, and 78,143 samples in the test set. Each sequence in the training and validation sets covers a 5-second window, sampled at a frequency of 10 Hz, providing dense temporal data for motion analysis. For the test set, only the first 2 seconds of the trajectories are publicly available, aligning with the Argoverse Motion Forecasting Challenge, which tasks participants with predicting the future 3-second trajectories of agents based on these initial 2-second observations.

This dataset setup is particularly advantageous for evaluating motion prediction models, as it encompasses diverse driving conditions and complex interactions among multiple agents. The availability of high-definition map data further enhances the richness of the dataset, allowing models to leverage detailed environmental context for improved trajectory forecasting.

Evaluation Metrics

To assess the effectiveness of our model, we utilise a set of well-established metrics commonly used in the field of motion prediction: minimum Average Displacement Error (minADE), minimum Final Displacement Error (minFDE), and Miss

Rate (MR). These metrics are designed to evaluate the accuracy and reliability of predicted trajectories over a defined forecast horizon, supporting the generation of up to six potential future paths for each agent.

Minimum Average Displacement Error (minADE): This metric measures the L_2 distance in meters between the best-predicted trajectory and the ground-truth trajectory, averaged over all future time steps. It provides an aggregate measure of how closely the predicted trajectory follows the actual path of the agent. The formula is given as:

$$\text{minADE} = \frac{1}{N} \sum_{i=1}^{N} \min_{k} \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{p}_{i,t}^{\text{pred},k} - \mathbf{p}_{i,t}^{\text{gt}}\|_2 \tag{6.29}$$

where:

- N: Total number of agents.
- T: Prediction horizon (number of future time steps).
- $\mathbf{p}_{i,t}^{\text{pred},k}$: Predicted position of agent i at time t for mode k.
- $\mathbf{p}_{i,t}^{\text{gt}}$: Ground-truth position of agent i at time t.
- k: Represents the mode of the multimodal prediction.

Minimum Final Displacement Error (minFDE): Unlike minADE, this metric focuses on the prediction error at the final predicted time step. It calculates the L_2 distance between the endpoint of the best-predicted trajectory and the ground-truth endpoint, offering insights into the model's ability to accurately predict the final position of an agent. The formula is given as:

$$\text{minFDE} = \frac{1}{N} \sum_{i=1}^{N} \min_{k} \|\mathbf{p}_{i,T}^{\text{pred},k} - \mathbf{p}_{i,T}^{\text{gt}}\|_2 \tag{6.30}$$

where:

- N: Total number of agents.
- T: Final predicted time step.
- $\mathbf{p}_{i,T}^{\text{pred},k}$: Predicted endpoint position of agent i for mode k.
- $\mathbf{p}_{i,T}^{\text{gt}}$: Ground-truth endpoint position of agent i.
- k: Represents the mode of the multimodal prediction.

Miss Rate (MR): This metric is defined as the percentage of scenarios where the distance between the ground-truth endpoint and the best-predicted endpoint exceeds 2.0 meters. It serves as an indicator of the model's capability to predict trajectories within a reasonable range of error. The formula is given as:

$$\text{MR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left(\min_{k} \|\mathbf{p}_{i,T}^{\text{pred},k} - \mathbf{p}_{i,T}^{\text{gt}}\|_2 > D_{\text{th}} \right) \tag{6.31}$$

where:

- N: Total number of agents.
- T: Final predicted time step.
- D_{th}: Threshold distance (e.g., 2.0 meters).
- $\mathbb{1}(\cdot)$: Indicator function that equals 1 if the condition is true, and 0 otherwise.
- k: Represents the mode of the multimodal prediction.

These metrics collectively provide a comprehensive evaluation of the model's performance, allowing for a detailed comparison with other state-of-the-art methods in motion prediction.

6.4.2 Implementation Details

Our model is implemented and trained on an NVIDIA RTX 3080 Ti GPU to leverage its high computational power. We employ the AdamW optimizer for training, which is well-suited for handling the sparse gradients typical in deep learning tasks. The training process spans 50 epochs, with a batch size of 32 to balance between convergence speed and memory efficiency.

The initial learning rate is set to 10^{-4}, with a weight decay of 1×10^{-2} to prevent overfitting by penalizing large weights. To further enhance training stability, we apply a dropout rate of 0.1 across the network layers. The learning rate is adjusted throughout the training process using a cosine annealing scheduler, which gradually reduces the learning rate to facilitate fine-tuning in later stages of training.

Our model consists of multiple layers tailored to capture both localised and global features. Specifically, the context encoder is composed of 4 layers of localised attention to focus on immediate interactions, while the temporal encoder includes 4 layers to effectively model temporal dependencies. The global contextual aggregator module is implemented with 3 layers to integrate broader scene information. Each decoder layer is equipped with 128 hidden units and utilises 8 attention heads in each multi-head attention block to capture diverse features.

For generating anchored goal queries, we use a Gaussian Mixture Model (GMM) to cluster potential intention points extracted from the training dataset. This approach provides more refined clustering compared to traditional k-means clustering, improving the representation of potential future positions.

6.4.3 Results

Performance Comparison with State-of-the-art: Table 6.1 presents a detailed comparison of our proposed model against various state-of-the-art methods, eval-

uated on the Argoverse validation set using standard metrics such as minADE, minFDE, and MR. Our model achieves competitive results with a minADE of 0.724, minFDE of 1.11, and MR of 0.11.

Table 6.1: Comparison of Performance Metrics Across Different Methods.

Method	minADE	minFDE	MR
LaneGCN Liang et al. (2020)	0.71	1.08	0.10
DenseTNT(w/100ms opt.) Gu et al. (2021)	0.73	1.05	0.10
DenseTNT(w/goal set pred.) Gu et al. (2021)	0.75	1.05	0.10
mmTransformer Liu et al. (2021)	0.713	1.153	0.106
HOME Gilles et al. (2021)	0.93	1.28	6.8
GOHOME Gilles et al. (2022)	0.8904	1.26	7.1
TPCN Ye et al. (2021)	0.73	1.15	0.11
HiVT-64 Zhou et al. (2022)	0.69	1.04	0.10
HiVT-128 Zhou et al. (2022)	0.66	0.96	0.09
Ours	0.724	1.11	0.11

Our approach performs slightly better than mmTransformer Liu et al. (2021), which has a minADE of 0.713, minFDE of 1.153, and MR of 0.106, indicating that our model can offer comparable prediction accuracy. Although HiVT-128 Zhou et al. (2022) demonstrates the best overall performance with a minADE of 0.66, minFDE of 0.96, and MR of 0.09, our model still maintains strong results, particularly given its relative simplicity compared to the HiVT architecture.

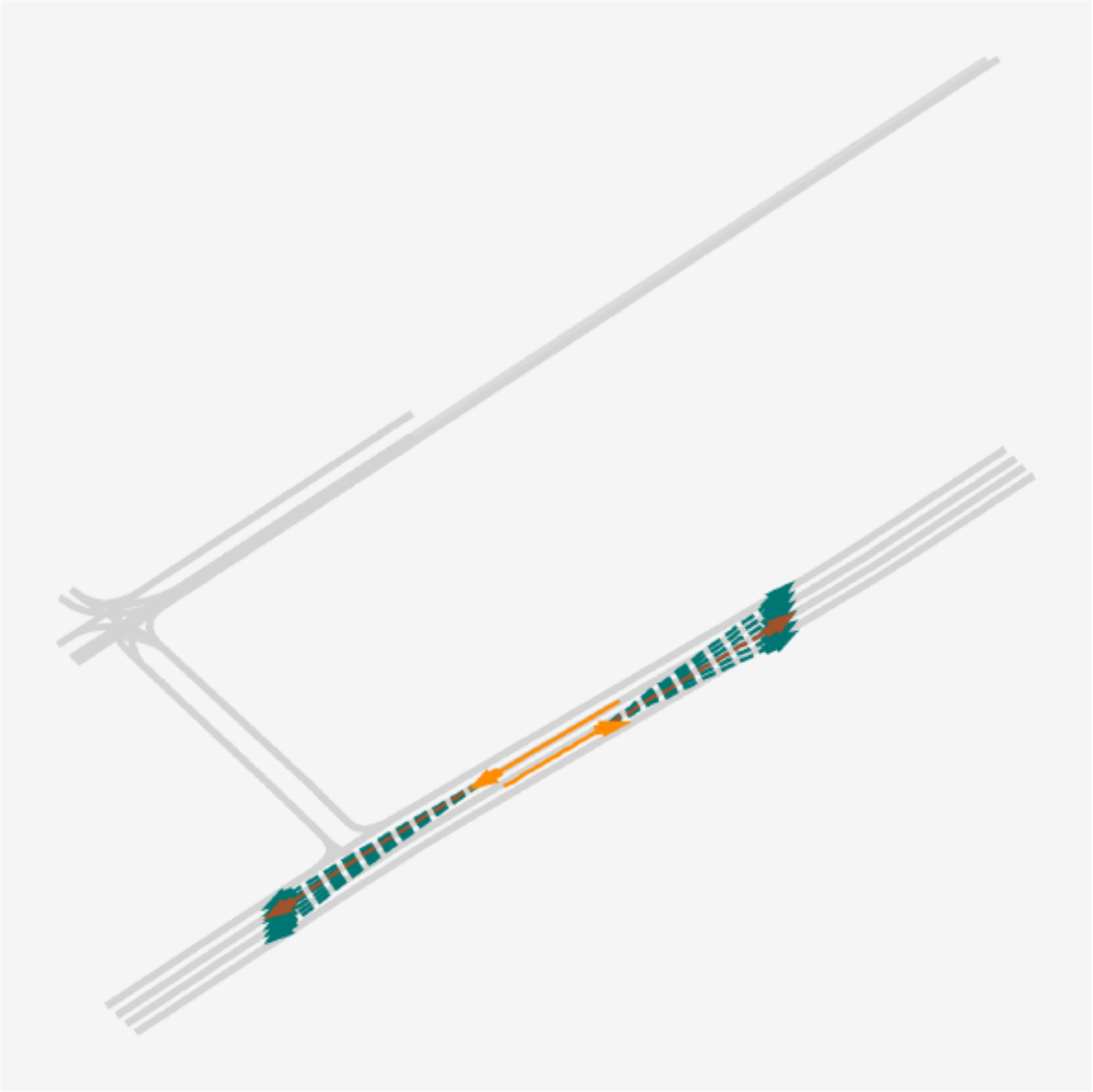

Figure 6.5: Qualitative result of IRMTR 1.

Moreover, our model shows significant improvements over models like GOHOME Gilles et al. (2022), which reports much higher minADE and minFDE values (0.8904 and 1.26, respectively) and a substantially higher MR of 7.1. The results also indicate that our model outperforms LaneGCN Liang et al. (2020) and remains highly competitive with DenseTNT's variations, which also show strong performance but with slightly different optimization setups.

Figure 6.6: Qualitative result of IRMTR 2.

Overall, the results confirm that our intention-refined transformer model effectively reduces prediction error and stands as a competitive method among the top-performing models in trajectory prediction tasks on the Argoverse validation set. Some qualitative results can be observed in Figure 6.5, Figure 6.6, Figure 6.7, and Figure 6.8.

Figure 6.7: Qualitative result of IRMTR 3.

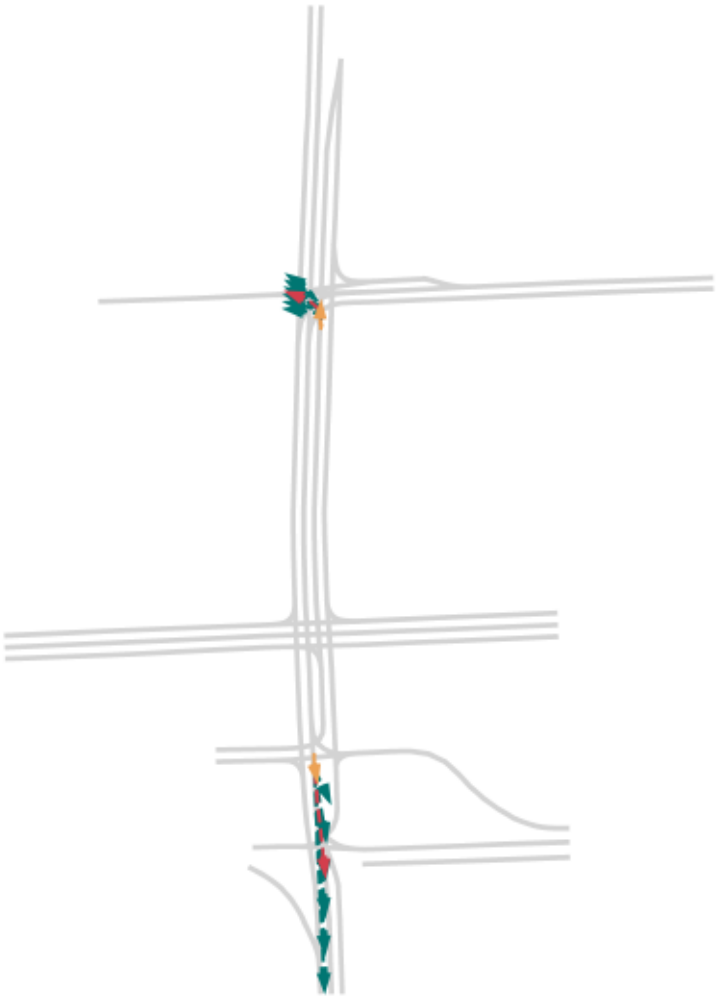

Figure 6.8: Qualitative result of IRMTR 4.

6.4.4 Ablation Studies

Importance of Each Module

We conduct a series of ablation studies to evaluate the impact of each module within our model, as shown in Table 6.2. This analysis focuses on the Temporal, Global, and Agent-Environment modules, highlighting their contributions to

overall model performance.

- **Temporal Module:** Removing the Temporal module results in a significant performance drop, with minADE and minFDE increasing to 1.09 and 1.64, respectively, and MR rising to 0.24. These findings emphasize the importance of temporal information in refining trajectory predictions and maintaining accuracy over time.
- **Global Module:** Excluding the Global module also leads to a noticeable degradation in performance, with minADE increasing to 0.77 and minFDE to 1.21, along with an MR of 0.13. This highlights the module's role in capturing long-range dependencies and enhancing the overall prediction quality.
- **Agent-Environment Module:** The removal of the Agent-Environment interaction module results in further performance degradation, with minADE increasing to 0.82, minFDE to 1.31, and MR to 0.15. This underscores the importance of modeling interactions between agents and their environment for accurate and reliable predictions.

Table 6.2: Ablation Study: Effect of Removing Key Components

Temporal	Global	Agent-En	minADE	minFDE	MR
✓	✓	✓	0.724	1.11	0.11
×	✓	✓	1.09	1.64	0.24
✓	×	✓	0.77	1.21	0.13
✓	✓	×	0.82	1.31	0.15

These ablation studies confirm that each component contributes significantly to the model's performance, with their combined use leading to the most accurate predictions, as indicated by the lowest minADE, minFDE, and MR values in the full model.

Importance of the Anchored Goal Query

An additional ablation study, shown in Table 6.3, investigates the effect of incorporating Anchored Goal Queries into our model. The results clearly demonstrate that these queries significantly enhance the model's performance.

When Anchored Goal Queries are utilised, the model achieves a minADE of 0.724 and a minFDE of 1.11, with an MR of 0.11. These values represent the best performance, indicating that Anchored Goal Queries help reduce uncertainty in future trajectory predictions, thereby improving both accuracy and reliability.

In contrast, removing the Anchored Goal Queries leads to a decline in performance, with minADE increasing to 0.741, minFDE to 1.23, and MR to 0.13. This decline highlights the importance of Anchored Goal Queries in guiding the model towards more accurate and plausible trajectory predictions.

Overall, these results from the ablation study demonstrate that Anchored Goal Queries are a crucial component for enhancing the prediction accuracy of our model.

Attention Mechanisms Difference

To explore the impact of different attention mechanisms on model performance, we conduct an ablation study as presented in Table 6.4. This study compares the use of global attention with localised attention in the model's architecture.

- **Global Attention:** When only global attention is used, the model achieves

Table 6.3: Ablation Study: Impact of Anchored Goal Query

Anchored Goal Query	minADE	minFDE	MR
✓	0.724	1.11	0.11
×	0.741	1.23	0.13

a minADE of 0.74 and a minFDE of 1.22, with an MR of 0.14. While these results are competitive, they suggest that relying solely on global interactions may overlook some finer, localised details that are crucial for accurate trajectory prediction.

- **Localised Attention:** In contrast, using localised attention results in superior performance with a minADE of 0.731, minFDE of 1.14, and an MR of 0.12. These improvements highlight the effectiveness of localised attention in capturing relevant spatial details, which contribute to more precise trajectory predictions.

Table 6.4: Ablation Study: Impact of Different Attention Mechanisms

Global Attention	Localised Attention	minADE	minFDE	MR
✓	×	0.74	1.22	0.14
×	✓	0.731	1.14	0.12

The findings from this ablation study demonstrate that localised attention is a more effective approach within our model, contributing to enhanced predictive

performance by focusing on critical local interactions that global attention alone may miss.

6.5 Conclusion

In this chapter, we proposed the Intention Refined Motion Transformer (IRMTR), an innovative architecture designed to address the complex task of multi-agent trajectory prediction in dynamic environments. By effectively balancing both local and global contextual information, IRMTR enhances the accuracy and robustness of motion forecasting. To achieve this balance, we integrated two key components into our framework: anchored goal queries and localised attention mechanisms.

Anchored goal queries play a pivotal role in guiding the model toward plausible future positions of agents, effectively narrowing down the search space for potential trajectories. By focusing the model's attention on critical areas of the scene, this mechanism enhances its ability to anticipate the likely paths agents might follow. Leveraging predefined intention points derived from historical data, anchored goal queries provide a structured approach to capturing diverse motion patterns, which is essential for generating accurate and reliable trajectory predictions. This strategy sets our model apart by efficiently incorporating probable future positions into the prediction process.

Simultaneously, the localised attention mechanisms enable the model to dynamically allocate computational resources to the most relevant spatial interactions within the environment. This targeted focus allows the model to capture fine-grained details and local dependencies crucial for understanding complex agent interactions. By concentrating on specific regions pertinent to agents' future movements, the localised attention mechanisms enhance the model's sensitivity to subtle cues and environmental variations, thereby contributing to more accu-

rate trajectory predictions. This approach offers a significant improvement over traditional global attention mechanisms by reducing complexity.

Through extensive experiments conducted on the Argoverse motion forecasting dataset, we have demonstrated that IRMTR significantly improves the prediction accuracy over existing baseline models. The model excels in reducing prediction errors across a variety of driving scenarios, including those with complex agent interactions and diverse environmental conditions. This capability is particularly important for real-world applications, where the ability to accurately forecast the movements of multiple agents can significantly enhance the safety and efficiency of autonomous driving systems.

The results of our ablation studies further underscore the importance of each component within our architecture. By systematically evaluating the impact of removing key modules, we have confirmed that both the anchored goal queries and localised attention mechanisms are integral to the model's overall performance. These components not only contribute individually to improving prediction accuracy but also work synergistically to refine the model's understanding of the scene, enabling it to make more informed predictions under various conditions.

In summary, the Intention Refined Motion Transformer represents a significant advancement in the field of trajectory prediction. The proposed architecture offers a robust and flexible framework that effectively integrates both local and global context to generate accurate and contextually aware predictions of agent trajectories. As a result, IRMTR not only enhances the current state-of-the-art in motion forecasting but also provides a solid foundation for future research and development in this domain. Potential extensions of this work include adapting the model to more complex environments, incorporating additional sensor modalities, and exploring broader applications beyond autonomous driving, such as robotics and human motion analysis.

The insights gained from this chapter align with the overarching goals of this thesis, which emphasizes the development of robust and reliable perception systems for autonomous driving. By addressing the challenges of motion forecasting, this chapter builds upon the advancements in multi-object tracking and paves the way for future research in comprehensive motion understanding, linking seamlessly with the narrative of improving autonomous vehicle safety and capability.

CHAPTER 7

Conclusion

7.1 Overview

Most research over the past couple of decades has focused on autonomous driving to make vehicles that can compete in driving safely through complicated environments without human assistance. The ability of the vehicle to perceive the environment correctly and predict the future motion of the other road users is the prime ability of this system. While considerable work has been done in this domain, a few challenges persist, particularly in the integration of diverse sensor data and the application of an advanced machine learning model to a real-world scenario.

Perception tasks in self-driving vehicles have traditionally relied on LiDAR and camera sensors. These sensors represent a trade-off between their merits and their demerits: in most cases, LiDAR enforces highly accurate depth but is generally very expensive, whereas its performance degrades based on the environmental conditions. Whilst cameras have richer visual detail, but suffers from depth perception and object recognition challenges in changing light conditions. Recent advancements have introduced synthetic data generation and deep learning models like

ViTs. These recent innovations also introduce their problems to efficiency of computation and the capability to generalize across environments.

Aiming to alleviate these challenges and seize the opportunities, this thesis aimed at improving recognition and prediction competence in autonomous driving with a series of innovative methodologies. It contributed to solving the problem of data scarcity in LiDAR segmentation by integrating synthetic data with real-world datasets. Our experiments showed that models trained on the integrated dataset achieved significant improvements in segmentation accuracy compared to those trained solely on real-world data, thus effectively improving model generalizability across different driving scenarios.

Besides, another contribution is adapting ViTs to LiDAR data processing. We transform 3D point cloud into a well-structured 2D format with the spherical projection technique to make it possible to utilize ViTs for complex spatial dependencies modeling. This way, it significantly lifted the accuracy of segmentation and increased robustness against diverse driving conditions.

We also proposed a multi-object tracking framework that utilized both 2D and 3D data through transformer-based sensor fusion. This work improved tracking performance and is also robust against occlusions and identity switches-providing insight into adapting transformer architectures for handling multi-object tracking in dynamic scenes.

Additionally, our research introduces the Intention Refined Motion Transformer and is a novel architecture. IRMTR combines the best features of intention-based prediction mechanisms with a hierarchical attention structure that helps to improve further the performance of trajectory prediction in complex driving scenarios. IRMTR reduced the prediction error in these uncertain, multi-agent interaction scenarios, thus contributing a new approach toward motion forecasting.

Among those contributions, some of the main challenges in perception, track-

ing, and prediction have been addressed in this thesis, with further improvement on the capability of autonomous driving systems. Through extensive experimentation and validation on several datasets and scenarios, we have shown that our methods can achieve salient improvements compared to the current techniques, confirming that our approaches are effective.

7.2 Future Work

While this thesis has made substantial progress in advancing the field of autonomous driving, there are several avenues for future research that could further enhance the capabilities of the proposed methodologies.

One potential direction is to expand the integration of synthetic data to other sensor modalities, such as radar or thermal cameras, to provide additional robustness to perception models. By diversifying the types of synthetic data used, future studies could explore the benefits of multi-sensor fusion in a wider range of driving conditions, particularly in scenarios where certain sensors may be compromised.

Other future directions include optimizing the Vision Transformer-based segmentation framework toward real-time data processing and online learning. It will be necessary to devise strategies that reduce computational complexity with no penalty on the accuracy of segmentations, hence enhancing adaptability within dynamic environments and making such models more feasible for application in real-world scenarios.

Another promising avenue is the advancement of a transformer-based sensor fusion and tracking framework that incorporates real-time data processing with online learning to add robustness in countering unprecedented situations with dynamically changing environmental factors. This would be further enhanced if algorithms could be designed with efficiency, adapting to new information on the

fly.

The Intention Refined Motion Transformer (IRMTR) offers significant potential for further enhancement and adaptation. This becomes further useful on investigation into its application in highly crowded urban areas or situations involving complicated scenarios of various modes of transport: bicycles, scooters, and pedestrians. A more efficient framework for predicting multimodal trajectories of multiple agents simultaneously would reduce redundant context encoding and enhance computational efficiency. Beyond rule-based post-processing, refining the selection of trajectories stands to offer a more robust and versatile framework that optimizes performance on key metrics like minADE and minFDE. Also, the incorporation of sophisticated intention recognition models or inclusions of feedback from real-time V2V communications would potentially enhance predictiveness and adaptability.

With respect to the future directions outlined above, we would like to build upon what this thesis has established as a foundation and, in doing so, contribute to the ongoing development of much safer and more efficient autonomous vehicles that could successfully navigate the complexities of real-world driving environments.

CHAPTER 8

Bibliography

Aksoy, E. E., Baci, S. & Cavdar, S. (2020), Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving, *in* '2020 IEEE intelligent vehicles symposium (IV)', IEEE, pp. 926–932.

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. & Savarese, S. (2016), Social lstm: Human trajectory prediction in crowded spaces, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 961–971.

Alonso, I., Riazuelo, L., Montesano, L. & Murillo, A. C. (2020), '3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation', *IEEE Robotics and Automation Letters* 5(4), 5432–5439.

Andriyenko, A. & Schindler, K. (2011), Multi-target tracking by continuous energy minimization, *in* 'CVPR 2011', IEEE, pp. 1265–1272.

Ba, J. L., Kiros, J. R. & Hinton, G. E. (2016), 'Layer normalization', *arXiv preprint arXiv:1607.06450* .

Bahdanau, D., Cho, K. & Bengio, Y. (2014), 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473* .

Bansal, M., Krizhevsky, A. & Ogale, A. (2018), 'Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst', *arXiv preprint arXiv:1812.03079* .

Bao, H., Dong, L., Piao, S. & Wei, F. (2021), 'Beit: Bert pre-training of image transformers', *arXiv preprint arXiv:2106.08254* .

Baser, E., Balasubramanian, V., Bhattacharyya, P. & Czarnecki, K. (2019), Fantrack: 3d multi-object tracking with feature association network, *in* '2019 IEEE Intelligent Vehicles Symposium (IV)', IEEE, pp. 1426–1433.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C. & Gall, J. (2019), Semantickitti: A dataset for semantic scene understanding of lidar sequences, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision', pp. 9297–9307.

Berclaz, J., Fleuret, F., Turetken, E. & Fua, P. (2011), 'Multiple object tracking using k-shortest paths optimization', *IEEE transactions on pattern analysis and machine intelligence* **33**(9), 1806–1819.

Bergmann, P., Meinhardt, T. & Leal-Taixe, L. (2019), Tracking without bells and whistles, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 941–951.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X. & Tian, Q. (2022), 'Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast', *arXiv preprint arXiv:2211.02556* .

Bishop, C. M. & Nasrabadi, N. M. (2006), *Pattern recognition and machine learning*, Vol. 4.

Brasó, G. & Leal-Taixé, L. (2020), Learning a neural solver for multiple object tracking, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 6247–6257.

Brostow, G. J., Fauqueur, J. & Cipolla, R. (2009), 'Semantic object classes in video: A high-definition ground truth database', *Pattern Recognition Letters* **30**(2), 88–97.

Brown, T. B. (2020), 'Language models are few-shot learners', *arXiv preprint ArXiv:2005.14165* .

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. & Beijbom, O. (2020), nuscenes: A multimodal dataset for autonomous driving, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 11621–11631.

Camuffo, E., Mari, D. & Milani, S. (2022), 'Recent advancements in learning algorithms for point clouds: An updated overview', *Sensors* **22**(4), 1357.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. & Zagoruyko, S. (2020), End-to-end object detection with transformers, *in* 'European conference on computer vision', Springer, pp. 213–229.

Casas, S., Luo, W. & Urtasun, R. (2018), Intentnet: Learning to predict intention from raw sensor data, *in* 'Conference on Robot Learning', PMLR, pp. 947–956.

Chai, Y., Sapp, B., Bansal, M. & Anguelov, D. (2019), 'Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction', *arXiv preprint arXiv:1910.05449* .

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese,

S., Savva, M., Song, S., Su, H. et al. (2015), 'Shapenet: An information-rich 3d model repository', *arXiv preprint arXiv:1512.03012* .

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D. et al. (2019), Argoverse: 3d tracking and forecasting with rich maps, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 8748–8757.

Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. (2017), 'Rethinking atrous convolution for semantic image segmentation', *arXiv preprint arXiv:1706.05587* .

Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S. & Urtasun, R. (2015), '3d object proposals for accurate object class detection', *Advances in neural information processing systems* **28**.

Chen, X., Ma, H., Wan, J., Li, B. & Xia, T. (2017), Multi-view 3d object detection network for autonomous driving, *in* 'Proceedings of the IEEE conference on Computer Vision and Pattern Recognition', pp. 1907–1915.

Chen, X., Milioto, A., Palazzolo, E., Giguere, P., Behley, J. & Stachniss, C. (2019), Suma++: Efficient lidar-based semantic slam, *in* '2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 4530–4537.

Cheng, R., Razani, R., Taghavi, E., Li, E. & Liu, B. (2021), 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 12547–12556.

Chollet, F. (2017), Xception: Deep learning with depthwise separable convolu-

tions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1251–1258.

Chu, Q., Ouyang, W., Li, H., Wang, X., Liu, B. & Yu, N. (2017), Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 4836–4845.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016), The cityscapes dataset for semantic urban scene understanding, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3213–3223.

Cortinhal, T., Tzelepis, G. & Aksoy, E. E. (2020), 'Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving', *arXiv preprint arXiv:2003.03653* .

Cui, H., Radosavljevic, V., Chou, F.-C., Lin, T.-H., Nguyen, T., Huang, T.-K., Schneider, J. & Djuric, N. (2019), Multimodal trajectory predictions for autonomous driving using deep convolutional networks, *in* '2019 international conference on robotics and automation (icra)', IEEE, pp. 2090–2096.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* .

Djuric, N., Radosavljevic, V., Cui, H., Nguyen, T., Chou, F.-C., Lin, T.-H., Singh, N. & Schneider, J. (2020), Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving, *in* 'Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision', pp. 2095–2104.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. (2020), 'An image is worth 16x16 words: Transformers for image recognition at scale', *arXiv preprint arXiv:2010.11929* .

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J. & Zisserman, A. (2015), 'The pascal visual object classes challenge: A retrospective', *International journal of computer vision* **111**, 98–136.

Fan, L., Xiong, X., Wang, F., Wang, N. & Zhang, Z. (2021), Rangedet: In defense of range view for lidar-based 3d object detection, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 2918–2927.

Fang, L., Jiang, Q., Shi, J. & Zhou, B. (2020), Tpnet: Trajectory proposal network for motion prediction, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 6797–6806.

Feichtenhofer, C., Pinz, A. & Zisserman, A. (2017), Detect to track and track to detect, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 3038–3046.

Frossard, D. & Urtasun, R. (2018), End-to-end learning of multi-sensor 3d tracking by detection, *in* '2018 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 635–642.

Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C. & Schmid, C. (2020), Vectornet: Encoding hd maps and agent dynamics from vectorized representation, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 11525–11533.

Geiger, A., Lenz, P., Stiller, C. & Urtasun, R. (2013), 'Vision meets robotics: The kitti dataset', *The International Journal of Robotics Research* **32**(11), 1231–1237.

Geiger, A., Lenz, P. & Urtasun, R. (2012), Are we ready for autonomous driving? the kitti vision benchmark suite, *in* '2012 IEEE conference on computer vision and pattern recognition', IEEE, pp. 3354–3361.

Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B. & Moutarde, F. (2021), Home: Heatmap output for future motion estimation, *in* '2021 IEEE International Intelligent Transportation Systems Conference (ITSC)', IEEE, pp. 500–507.

Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B. & Moutarde, F. (2022), Gohome: Graph-oriented heatmap output for future motion estimation, *in* '2022 international conference on robotics and automation (ICRA)', IEEE, pp. 9107–9114.

Giuliari, F., Hasan, I., Cristani, M. & Galasso, F. (2021), Transformer networks for trajectory forecasting, *in* '2020 25th international conference on pattern recognition (ICPR)', IEEE, pp. 10335–10342.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT press.

Gu, J., Sun, C. & Zhao, H. (2021), Densetnt: End-to-end trajectory prediction from dense goal sets, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision', pp. 15303–15312.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S. & Alahi, A. (2018), Social gan: Socially acceptable trajectories with generative adversarial networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2255–2264.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017), Mask r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 2961–2969.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1026–1034.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012), 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:1207.0580* .

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* 9(8), 1735–1780.

Hong, J., Sapp, B. & Philbin, J. (2019), Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 8454–8462.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N. & Markham, A. (2020), Randla-net: Efficient semantic segmentation of large-scale point clouds, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 11108–11117.

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. (2017), Densely connected convolutional networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4700–4708.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. & Keutzer, K. (2016), 'Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size', *arXiv preprint arXiv:1602.07360* .

Inan, B. A., Rondao, D. & Aouf, N. (2023), Enhancing lidar point cloud segmentation with synthetic data, *in* '2023 31st Mediterranean Conference on Control and Automation (MED)', IEEE, pp. 370–375.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', pmlr, pp. 448–456.

Jaritz, M., Vu, T.-H., Charette, R. d., Wirbel, E. & Pérez, P. (2020), xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 12605–12614.

Kendall, A., Badrinarayanan, V. & Cipolla, R. (2015), 'Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding', *arXiv preprint arXiv:1511.02680* .

Keuper, M., Tang, S., Andres, B., Brox, T. & Schiele, B. (2018), 'Motion segmentation & multiple object tracking by correlation co-clustering', *IEEE transactions on pattern analysis and machine intelligence* **42**(1), 140–153.

Kim, A., Ošep, A. & Leal-Taixé, L. (2021), Eagermot: 3d multi-object tracking via sensor fusion, *in* '2021 IEEE International conference on Robotics and Automation (ICRA)', IEEE, pp. 11315–11321.

Kingma, D. P. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kochanov, D., Nejadasl, F. K. & Booij, O. (2020), 'Kprnet: Improving projection-based lidar semantic segmentation', *arXiv preprint arXiv:2007.12668* .

Kong, L., Liu, Y., Chen, R., Ma, Y., Zhu, X., Li, Y., Hou, Y., Qiao, Y. & Liu, Z. (2023), Rethinking range view representation for lidar segmentation, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision', pp. 228–240.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**.

Lai, X., Chen, Y., Lu, F., Liu, J. & Jia, J. (2023), Spherical transformer for lidar-based 3d recognition, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 17545–17555.

Landrieu, L. & Simonovsky, M. (2018), Large-scale point cloud semantic segmentation with superpoint graphs, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4558–4567.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J. & Beijbom, O. (2019), Pointpillars: Fast encoders for object detection from point clouds, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 12697–12705.

Le Cun, Y., Jackel, L., Boser, B., Denker, J., Graf, H. & GuyOI, I. (1989), 'Handwritten digit recognition: Applications of neural network chips'.

Li, L. L., Yang, B., Liang, M., Zeng, W., Ren, M., Segal, S. & Urtasun, R. (2020), End-to-end contextual perception and prediction with interaction transformer, *in* '2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 5784–5791.

Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S. & Urtasun, R. (2020), Learning lane graph representations for motion forecasting, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16', Springer, pp. 541–556.

Liu, Y., Zhang, J., Fang, L., Jiang, Q. & Zhou, B. (2021), Multimodal motion prediction with stacked transformers, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 7577–7586.

Long, J., Shelhamer, E. & Darrell, T. (2015), Fully convolutional networks for semantic segmentation, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3431–3440.

Maas, A. L., Hannun, A. Y., Ng, A. Y. et al. (2013), Rectifier nonlinearities improve neural network acoustic models, *in* 'Proc. icml', Vol. 30, Atlanta, GA, p. 3.

Macqueen, J. (1967), Some methods for classification and analysis of multivariate observations, *in* 'Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press'.

Maddern, W., Pascoe, G., Linegar, C. & Newman, P. (2017), '1 year, 1000 km: The oxford robotcar dataset', *The International Journal of Robotics Research* **36**(1), 3–15.

Mangalam, K., Girase, H., Agarwal, S., Lee, K.-H., Adeli, E., Malik, J. & Gaidon, A. (2020), It is not the journey but the destination: Endpoint conditioned trajectory prediction, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16', Springer, pp. 759–776.

Maturana, D. & Scherer, S. (2015), Voxnet: A 3d convolutional neural network for real-time object recognition, *in* '2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)', IEEE, pp. 922–928.

Mercat, J., Gilles, T., El Zoghby, N., Sandou, G., Beauvois, D. & Gil, G. P. (2020), Multi-head attention for multi-modal joint vehicle motion forecasting, *in* '2020 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 9638–9644.

Milioto, A., Vizzo, I., Behley, J. & Stachniss, C. (2019), Rangenet++: Fast and accurate lidar semantic segmentation, *in* '2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)', IEEE, pp. 4213–4220.

Moosmann, F. & Stiller, C. (2013), Joint self-localization and tracking of generic objects in 3d range data, *in* '2013 IEEE International Conference on Robotics and Automation', IEEE, pp. 1146–1152.

Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT press.

Neuhold, G., Ollmann, T., Rota Bulo, S. & Kontschieder, P. (2017), The mapillary vistas dataset for semantic understanding of street scenes, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 4990–4999.

Ngiam, J., Caine, B., Vasudevan, V., Zhang, Z., Chiang, H.-T. L., Ling, J., Roelofs, R., Bewley, A., Liu, C., Venugopal, A. et al. (2021), 'Scene transformer: A unified architecture for predicting multiple agent trajectories', *arXiv preprint arXiv:2106.08417* .

Osep, A., Mehner, W., Mathias, M. & Leibe, B. (2017), Combined image-and world-space tracking in traffic scenes, *in* '2017 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1988–1995.

Ošep, A., Mehner, W., Voigtlaender, P. & Leibe, B. (2018), Track, then decide: Category-agnostic vision-based multi-object tracking, *in* '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3494–3501.

Phan-Minh, T., Grigore, E. C., Boulton, F. A., Beijbom, O. & Wolff, E. M. (2020), Covernet: Multimodal behavior prediction using trajectory sets, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 14074–14083.

Pirsiavash, H., Ramanan, D. & Fowlkes, C. C. (2011), Globally-optimal greedy algorithms for tracking a variable number of objects, *in* 'CVPR 2011', IEEE, pp. 1201–1208.

Poudel, R. P., Liwicki, S. & Cipolla, R. (2019), 'Fast-scnn: Fast semantic segmentation network', *arXiv preprint arXiv:1902.04502* .

Press, O. & Wolf, L. (2016), 'Using the output embedding to improve language models', *arXiv preprint arXiv:1608.05859* .

Qi, C. R., Liu, W., Wu, C., Su, H. & Guibas, L. J. (2018), Frustum pointnets for 3d object detection from rgb-d data, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 918–927.

Qi, C. R., Su, H., Mo, K. & Guibas, L. J. (2017), Pointnet: Deep learning on point sets for 3d classification and segmentation, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 652–660.

Qi, C. R., Yi, L., Su, H. & Guibas, L. J. (2017), 'Pointnet++: Deep hierarchical feature learning on point sets in a metric space', *Advances in neural information processing systems* **30**.

Ren, S., He, K., Girshick, R. & Sun, J. (2015), 'Faster r-cnn: Towards real-time object detection with region proposal networks', *Advances in neural information processing systems* **28**.

Rhinehart, N., Kitani, K. M. & Vernaza, P. (2018), R2p2: A reparameterized push-forward policy for diverse, precise generative path forecasting, *in* 'Proceedings of the European Conference on Computer Vision (ECCV)', pp. 772–788.

Rhinehart, N., McAllister, R., Kitani, K. & Levine, S. (2019), Precog: Prediction conditioned on goals in visual multi-agent settings, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision', pp. 2821–2830.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D. & Lopez, A. M. (2016), The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3234–3243.

Rosu, R. A., Schütt, P., Quenzel, J. & Behnke, S. (2019), 'Latticenet: Fast point cloud segmentation using permutohedral lattices', *arXiv preprint arXiv:1912.05905* .

Roynard, X., Deschaud, J.-E. & Goulette, F. (2018), 'Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification', *The International Journal of Robotics Research* **37**(6), 545–557.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986', *Biometrika* **71**, 599–607.

Ruppel, F., Faion, F., Gläser, C. & Dietmayer, K. (2022), Transformers for multi-object tracking on point clouds, *in* '2022 IEEE Intelligent Vehicles Symposium (IV)', IEEE, pp. 852–859.

Russell, S. J. & Norvig, P. (2016), *Artificial intelligence: a modern approach*, Pearson.

Salzmann, T., Ivanovic, B., Chakravarty, P. & Pavone, M. (2020), Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16', Springer, pp. 683–700.

Sharma, S., Ansari, J. A., Murthy, J. K. & Krishna, K. M. (2018), Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking, *in* '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3508–3515.

Shi, S., Wang, X. & Li, H. (2019), Pointrcnn: 3d object proposal generation and detection from point cloud, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 770–779.

Shi, W. & Rajkumar, R. (2020), 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', pp. 1711–1719.

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H. & Kautz, J. (2018), Splatnet: Sparse lattice networks for point cloud processing, *in* 'Pro-

ceedings of the IEEE conference on computer vision and pattern recognition', pp. 2530–2539.

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

Sun, Q., Huang, X., Gu, J., Williams, B. C. & Zhao, H. (2022), M2i: From factored marginal trajectory prediction to interactive prediction, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 6543–6552.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014), 'Sequence to sequence learning with neural networks', *Advances in neural information processing systems* **27**.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1–9.

Tan, M. & Le, Q. (2019), Efficientnet: Rethinking model scaling for convolutional neural networks, *in* 'International conference on machine learning', PMLR, pp. 6105–6114.

Tang, C. & Salakhutdinov, R. R. (2019), 'Multiple futures prediction', *Advances in neural information processing systems* **32**.

Tatarchenko, M., Park, J., Koltun, V. & Zhou, Q.-Y. (2018), Tangent convolutions for dense prediction in 3d, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3887–3896.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J. & Savarese, S. (2017), Segcloud: Semantic segmentation of 3d point clouds, *in* '2017 international conference on 3D vision (3DV)', IEEE, pp. 537–547.

Teichman, A., Levinson, J. & Thrun, S. (2011), Towards 3d object recognition via classification of arbitrary object tracks, *in* '2011 IEEE International Conference on Robotics and Automation', IEEE, pp. 4034–4041.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F. & Guibas, L. J. (2019), Kpconv: Flexible and deformable convolution for point clouds, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 6411–6420.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A. & Jégou, H. (2021), Training data-efficient image transformers & distillation through attention, *in* 'International conference on machine learning', PMLR, pp. 10347–10357.

Turing, A. M. (1950), 'Mind', *Mind* **59**(236), 433–460.

Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K. S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C. P., Anguelov, D. et al. (2022), Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction, *in* '2022 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 7814–7821.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), 'Attention is all you need', *Advances in neural information processing systems* **30**.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A. & Leibe, B. (2019), Mots: Multi-object tracking and segmentation, *in* 'Proceedings of

the ieee/cvf conference on computer vision and pattern recognition', pp. 7942–7951.

Wang, J., Xu, H., Narasimhan, M. & Wang, X. (2021), 'Multi-person 3d motion prediction with multi-range transformers', *Advances in Neural Information Processing Systems* **34**, 6036–6049.

Wang, X., Girshick, R., Gupta, A. & He, K. (2018), Non-local neural networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 7794–7803.

Wang, Y., Kitani, K. & Weng, X. (2021), Joint object detection and multi-object tracking with graph neural networks, *in* '2021 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 13708–13715.

Weng, X., Wang, J., Held, D. & Kitani, K. (2020), 3d multi-object tracking: A baseline and new evaluation metrics, *in* '2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 10359–10366.

Weng, X., Wang, Y., Man, Y. & Kitani, K. M. (2020), Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 6499–6508.

Wu, B., Wan, A., Yue, X. & Keutzer, K. (2018), Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud, *in* '2018 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 1887–1893.

Wu, B., Zhou, X., Zhao, S., Yue, X. & Keutzer, K. (2019), Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmen-

tation from a lidar point cloud, *in* '2019 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 4376–4382.

Xiao, A., Yang, X., Lu, S., Guan, D. & Huang, J. (2021), 'Fps-net: A convolutional fusion network for large-scale lidar point cloud segmentation', *ISPRS Journal of Photogrammetry and Remote Sensing* **176**, 237–249.

Xu, C., Wu, B., Wang, Z., Zhan, W., Vajda, P., Keutzer, K. & Tomizuka, M. (2020), Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation, *in* 'European Conference on Computer Vision', Springer, pp. 1–19.

Xu, Y., Osep, A., Ban, Y., Horaud, R., Leal-Taixé, L. & Alameda-Pineda, X. (2020), How to train your deep multi-object tracker, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 6787–6796.

Yan, X., Gao, J., Zheng, C., Zheng, C., Zhang, R., Cui, S. & Li, Z. (2022), 2dpass: 2d priors assisted semantic segmentation on lidar point clouds, *in* 'European Conference on Computer Vision', Springer, pp. 677–695.

Yan, X., Zheng, C., Li, Z., Wang, S. & Cui, S. (2020), Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 5589–5598.

Ye, M., Cao, T. & Chen, Q. (2021), Tpcn: Temporal point cloud networks for motion forecasting, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 11318–11327.

Yin, T., Zhou, X. & Krahenbuhl, P. (2021), Center-based 3d object detection and tracking, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 11784–11793.

Yu, C., Ma, X., Ren, J., Zhao, H. & Yi, S. (2020), Spatio-temporal graph transformer networks for pedestrian trajectory prediction, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16', Springer, pp. 507–523.

Yuan, Y., Weng, X., Ou, Y. & Kitani, K. M. (2021), Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision', pp. 9813–9823.

Zeiler, M. D. & Fergus, R. (2014), Visualizing and understanding convolutional networks, *in* 'Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13', Springer, pp. 818–833.

Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X. & Wei, Y. (2022), Motr: End-to-end multiple-object tracking with transformer, *in* 'European Conference on Computer Vision', Springer, pp. 659–675.

Zeng, Y., Ma, C., Zhu, M., Fan, Z. & Yang, X. (2021), Cross-modal 3d object detection and tracking for auto-driving, *in* '2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3850–3857.

Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. (2021), 'Understanding deep learning (still) requires rethinking generalization', *Communications of the ACM* **64**(3), 107–115.

Zhang, R., Wu, Y., Jin, W. & Meng, X. (2023), 'Deep-learning-based point cloud semantic segmentation: A survey', *Electronics* **12**(17), 3642.

Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J. & Loy, C. C. (2019), Robust multi-

modality multi-object tracking, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 2365–2374.

Zhang, Y., Sheng, H., Wu, Y., Wang, S., Lyu, W., Ke, W. & Xiong, Z. (2020), 'Long-term tracking with deep tracklet association', *IEEE Transactions on Image Processing* **29**, 6694–6706.

Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B. & Foroosh, H. (2020), Polarnet: An improved grid representation for online lidar point clouds semantic segmentation, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 9601–9610.

Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C. et al. (2021), Tnt: Target-driven trajectory prediction, *in* 'Conference on Robot Learning', PMLR, pp. 895–904.

Zhao, H., Jiang, L., Jia, J., Torr, P. H. & Koltun, V. (2021), Point transformer, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 16259–16268.

Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. (2017), Pyramid scene parsing network, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2881–2890.

Zhou, X., Koltun, V. & Krähenbühl, P. (2020), Tracking objects as points, *in* 'European conference on computer vision', Springer, pp. 474–490.

Zhou, X., Wang, D. & Krähenbühl, P. (2019), 'Objects as points', *arXiv preprint arXiv:1904.07850* .

Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J.

& Vasudevan, V. (2020), End-to-end multi-view fusion for 3d object detection in lidar point clouds, *in* 'Conference on Robot Learning', PMLR, pp. 923–932.

Zhou, Y. & Tuzel, O. (2018), Voxelnet: End-to-end learning for point cloud based 3d object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4490–4499.

Zhou, Z., Ye, L., Wang, J., Wu, K. & Lu, K. (2022), Hivt: Hierarchical vector transformer for multi-agent motion prediction, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 8823–8833.

Zhu, J., Yang, H., Liu, N., Kim, M., Zhang, W. & Yang, M.-H. (2018), Online multi-object tracking with dual matching attention networks, *in* 'Proceedings of the European conference on computer vision (ECCV)', pp. 366–382.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X. & Dai, J. (2020), 'Deformable detr: Deformable transformers for end-to-end object detection', *arXiv preprint arXiv:2010.04159* .

Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H. & Lin, D. (2021), Cylindrical and asymmetrical 3d convolution networks for lidar segmentation, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 9939–9948.