Research internship report

3D Occupancy for Motion Forecasting

Akedjou Achraff ADJILEYE
Internship carried out from 11/04/2024 to 11/10/2024

Internship tutors : Yihong XU and Alexandre BOULCH
Referent teacher : Gül Varol

Educational institution : Master of Research, Mathématiques Vision Apprentissage[1] (MVA)
Internship host company : Valeo.ai - 100 Rue de Courcelles, 75017 Paris

[1]Mathematic Vision Learning

Acknowledgments

I would like to express my gratitude to **Ms VAROL**, who was the university supervisor for my internship.

I am also grateful to Valeo.ai for hosting me during this 6-month internship. I would also like to thank **Yihong XU** and **Alexandre BOULCH**, who supervised my internship, for making it possible under optimal conditions. I learned a lot from their advice, their experience, and their desire to pass on their knowledge. The support, freedom, autonomy, and responsibilities given to me enabled me to tackle research more effectively.

Many thanks to **Matthieu CORD** Scientific Director of Valeo.ai and to all the permanent researchers, PhDs, and interns of the Valeo.ai team for their warm welcome. In particular, I deeply appreciate the discussion with **Éloi ZABLOCKI** and **Lan FENG**[2]; they provided invaluable assistance in overcoming the challenges I encountered with the UniTraj Feng et al. (2024) framework, which was the foundation of my work throughout the internship.

[2]*EPFL, Switzerland

Global Context of the Internship

Introduction to Valeo.ai

Valeo.ai[3] is an international team based in Paris, composed of permanent researchers and PhD students, led by Matthieu Cord. The team is dedicated to conducting AI research for automotive applications at Valeo, collaborating closely with world-class academics. Our primary research focuses on developing advanced, clearer, and safer automotive AI. The Lab offer a research-oriented environment with robust hardware support, enabling interns to carry out cutting-edge research. This internship is aligned with the fields of 3D vision and motion forecasting within the context of autonomous driving.

Scope of the Internship Project

Autonomous driving technology has undergone remarkable progress in recent years. Figure 1 illustrates a typical pipeline used in autonomous driving, encompassing steps from sensor data acquisition to planning. Among these steps, motion forecasting is a crucial intermediary task that aims to predict the future trajectories of moving agents within the scene. Typically, the input to the motion forecasting module is multimodal, drawing on the outputs of the perception block, which utilizes various models to perform tasks such as detection, tracking, and mapping.

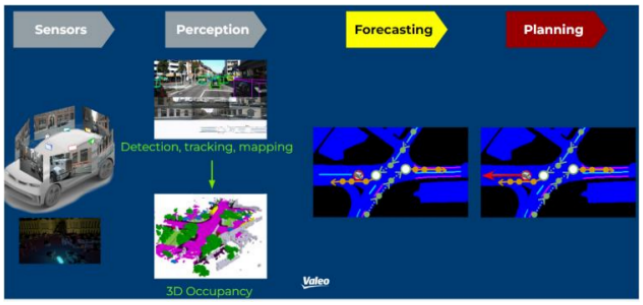


Figure 1: When 3D Occupancy meets motion forecasting

However, integrating these diverse perception tasks with motion forecasting is non-trivial and can introduce significant noise. In parallel, recent advancements in 3D occupancy tasks have enabled the generation of rich scene representations by predicting semantic labels for various objects—such as drivable surfaces, vehicles, pedestrians, and traffic cones—in a dense 3D space. This representation serves as a potential method for unifying the different inputs of motion forecasting into a single perception object with a unimodal input.

The internship project aims to explore the feasibility of combining motion forecasting and 3D occupancy—in an end-to-end manner through novel design approaches. The project investigates the impact of such integration on the performance of motion forecasting.

We explored several methods for integrating occupancy into a motion forecasting pipeline. Our research focused on evaluating how occupancy features map, obtained through the encoding of 3D point cloud provided by a LiDAR sensor, could potentially improve motion forecasting, and replace the curated symbolic inputs typically used in motion forecasting models, which are often expensive and

[3]https://www.valeo.com/en/valeo-ai/

time-consuming to obtain. We conducted extensive comparisons of our approach against state-of-the-art methods using a real-world dataset and demonstrated superior performance across several key metrics.

Key words: motion forecasting, LiDAR point cloud, occupancy features map, early fusion, late fusion.

Contents

1 Introduction **5**

2 Motion Forecasting Models **5**
2.1 Wayformer 5
2.1.1 Inputs 6
2.1.2 The Model 6
2.1.3 Motion Forecasting Evaluation Metrics 8
2.2 Wayformer in Practice: The UniTraj Framework 9
2.2.1 Dataset, Input Representation, and Encoding Strategy 9
2.2.2 Decoding Strategy 10
2.2.3 Experiments: Baseline Reproduction 11
2.3 Occupancy and Wayformer 11
2.3.1 Point Cloud Encoding 12
2.3.2 Decoding Strategy for Occupancy in Motion Forecasting 14
2.3.3 Experiments and Results 16

3 Related Work **18**
3.1 WOMD-LiDAR: A Raw Sensor Dataset Benchmark for Motion Forecasting 18
3.2 MoST: Multi-modality Scene Tokenization for Motion Forecasting 19
3.3 Limitations Compared to Our Work 20

4 Wayoccformer Ablation Study: Towards Occupancy-Based Motion Forecasting. **21**

5 Semantic Occupancy for Motion Forecasting **23**
5.1 Adding Semantic Labels in LiDAR Points Features 23

6 Conclusion and Future Works **25**

A The nuScenes Dataset, a standart dataset for autonomous driving. **29**
A.1 Dataset Processing 29

1 Introduction

In this section, we introduce the task of motion forecasting, highlighting his multi-modal aspect.

- **Motion Forecasting Models Have Multi-Modal Inputs**: As with any forecasting task, the setup of motion forecasting is straightforward: given the past trajectory of an agent[4] of interest in a driving scene, we want to predict the future trajectory. We explore data-driven solutions where we aim to learn a machine learning model $\mathcal{M}$ such that for an agent A in a driving scene, given its 2D past trajectory coordinates

  $$Y_{1:T}^{A} = \{(x_1, y_1)^A, \ldots, (x_T, y_T)^A\}$$

  and other additional inputs $\mathcal{X}_{1:T}^{A}$ that describe the driving scene during the past timestamps, we want to predict the future trajectory

  $$Y_{T+1:T_f}^{A} = \{(x_{T+1}, y_{T+1}), \ldots, (x_{T_f}, y_{T_f})\} = \mathcal{M}(Y_{1:T}^{A}, \mathcal{X}_{1:T}^{A}),$$

  where T is the number of past timestamps and T_f is the final future timestamp.

  $\mathcal{X}_{1:T}^{A}$ represents a diverse set of inputs. Depending on the MF datasets, it may contain trajectories and state information about surrounding vehicles and other types of agents (e.g., cyclists, pedestrians, static objects) in the driving scene, state information about traffic lights, raw sensor inputs like multi-view camera images or LiDAR point clouds, and road mapping data. MF models can therefore be considered multi-modal, depending on the inputs they incorporate.

- **Motion Forecasting Outputs Are Also Multi-Modal**: Since a vehicle can follow multiple possible trajectories on the road, the commonly adopted approach in the literature is to predict the future trajectory of an agent as a mixture of Gaussians. This mixture represents the possible trajectories, each with an associated probability score for each trajectory mode. MF models are trained to maximize the log-likelihood of selecting the most likely trajectory and the Gaussian log-probability of the ground truth trajectory.

2 Motion Forecasting Models

In this section, we explore various motion forecasting models, focusing primarily on Wayformer Nayakanti et al. (2022), a transformer-based model designed to integrate diverse input modalities for trajectory prediction. We will first outline the key components of Wayformer, including its input representation, model architecture, and evaluation metrics. Additionally, we will discuss how we utilized the UniTraj framework Feng et al. (2024) to reproduce Wayformer's results and conduct further experiments. Finally, we will introduce our methodology for incorporating occupancy information into Wayformer, demonstrating the effectiveness of this approach in improving motion forecasting performance.

2.1 Wayformer

Wayformer Nayakanti et al. (2022) proposes a modality-agnostic framework to integrate information from different modalities, including agent past trajectories, agent interactions, and road graphs, in a homogeneous way, enabling them to be encoded with a transformer-based network Vaswani et al. (2023).

[4]We focus on vehicle trajectory forecasting in this work, so unless explicitly mentioned otherwise, we will use the terms agent and vehicle interchangeably throughout this report.

2.1.1 Inputs

Each modality is represented as a **4**-dimensional (4D) tensor as follows:

Agent of interest past trajectories, a tensor of shape $[A, T, 1, D]$,

Context agent trajectories, a tensor of shape $[A, T, S, D]$, represents the other agents (not only vehicles) in the scene closest to the agents of interest,

Road graphs, a tensor of shape $[A, 1, S_r, D]$ that contains road features around the agents of interest,

Traffic Light State, a tensor of shape $[A, T, S_{tls}, D]$, which contains the state of the traffic signals closest to the agents of interest,

where A is the number of agents of interest (for which we want to forecast the trajectories), T is the number of past timestamps, S is the number of agents closest to the agents of interest, S_r is the number of road graph segments, S_{tls} is the number of traffic light signals, and D is the latent feature dimension of all the input modalities. A multi-layer perceptron (MLP) is used to project each modality's raw state features into a D-dimensional feature space, as different inputs may not share the same number of raw features. For example, for an agent's raw state features, we have the 2-dimensional (2D) position (x, y), velocity (v_x, v_y), acceleration (a_x, a_y), and a one-hot encoding of the agent category (only for context agents as they are not necessarily vehicles), while we only have the 2D position (x, y) for the map polyline points.

Each road graph segment is represented as polylines that approximate the road shape with a collection of line segments specified by their endpoints and annotated with type information (e.g., lanes, intersections, etc.).

2.1.2 The Model

Wayformer is a family of encoder-decoder models (see Figure 2).

The Encoder encodes the previous inputs using one or more attention encoders depending on the modality fusion strategy adopted. The authors proposed three fusion strategies, as described in Figure 3.

Late Fusion: In this approach, each modality has its own dedicated self-attention encoder. The width and depth of these encoders are the same, and the outputs are concatenated before the attention layers in the decoder.

Early Fusion: In this paradigm, the scene encoder consists of a single self-attention encoder (cross-modal encoder), giving the network maximum flexibility in assigning importance across modalities.

Hierarchical Fusion: As a compromise between the two previous extremes, capacity is split between modality-specific self-attention encoders and the cross-modal encoder in a hierarchical fashion. As in late fusion, the width and depth are common across the attention encoders and the cross-modal

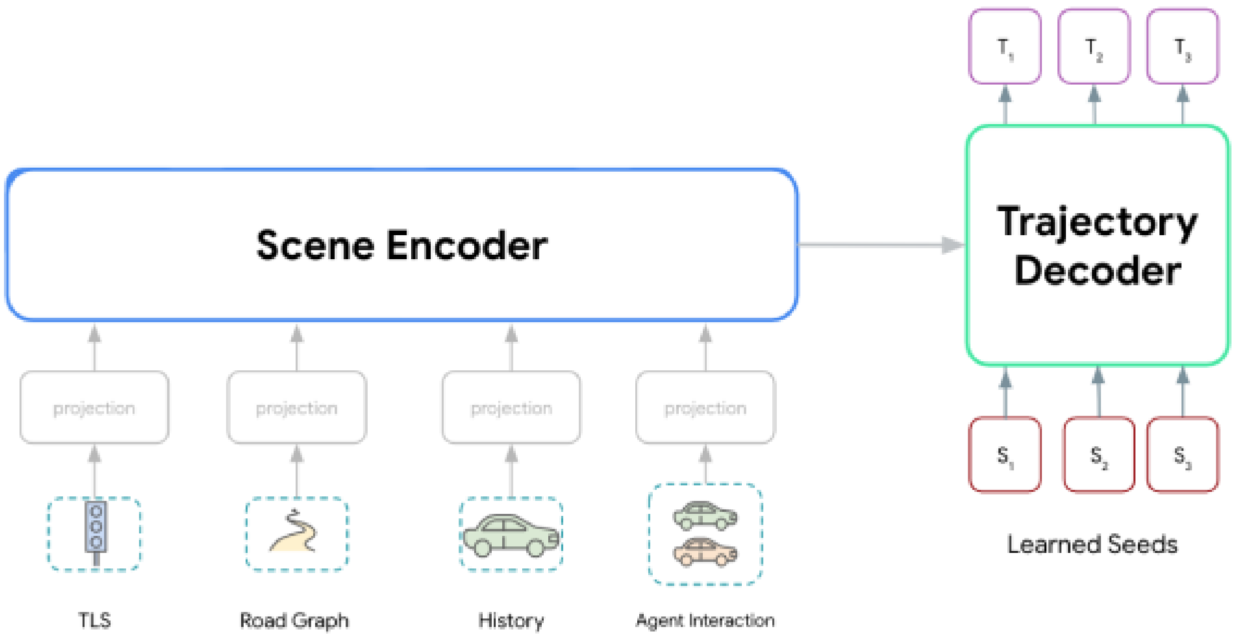


Figure 2: Wayformer overall architecture: the encoder encodes the different input modalities to represent the scene in a latent space, and the decoder uses some latent trajectory queries to decode the scene and output multiple trajectory modes per agent of interest.

encoder. This effectively divides the depth of the scene encoder between modality-specific encoders and the cross-modal encoder.

Self-attention is naturally permutation equivariant. However, for agent states across different time steps, which follow a specific ordering, it is beneficial to break permutation equivariance and utilize the sequence information. Wayformer then uses learned positional embeddings for all modalities. Given that not all modalities are ordered, the learned positional embeddings are initially set to zero, allowing the model to learn if it is necessary to utilize the ordering within a modality.

Attention Mechanism: In our reproduced experiments with Wayformer, we used the early fusion strategy, as it was claimed to be the best in Nayakanti et al. (2022). For each modality, we have a time dimension T_m and a spatial dimension S_m that are concatenated to form the number of tokens for the attention mechanism, a token being an agent in the scene at a given timestamp. We then feed an encoder layer with an input tensor X of shape $[A, N_m = S_m \times T_m, D]$ where A is the number of agents of interest at one training step (also the batch size), N_m is the number of tokens (a token being a modality state at a given timestamp), and D is the latent feature dimension. As N_m is large ($\sim 10^4$), Nayakanti et al. (2022) uses **latent query attention** to address the quadratic complexity cost of the attention mechanism in the first encoder block. Latent query attention maps X to a latent space $Z \in \mathcal{R}^{A,L_{out},D}$ with $L_{out} \ll N_m$. These latents Z are further processed through a series of encoder layers that take in and return arrays in the same latent space (see Figure 4).

The Decoder: Given the scene embedding $Z \in \mathcal{R}^{A,L_{out},D}$ from the encoder, the decoder is a

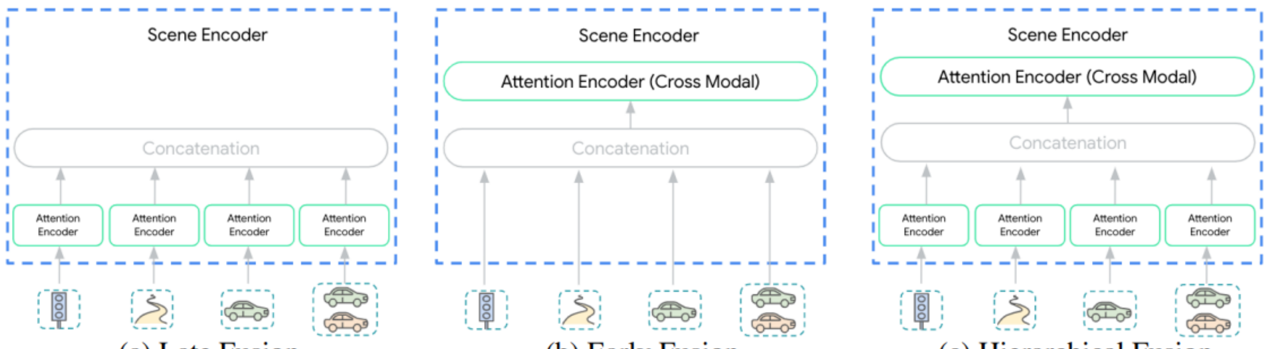


Figure 3: Wayformer encoder fusing multimodal inputs at different stages. Late fusion dedicates an attention encoder per modality, while early fusion processes all inputs within one cross-modal encoder. Finally, hierarchical fusion combines both approaches.

transformer network that is fed a set of k learned initial queries $(S_i \in \mathcal{R}^h)_{i=1}^k$; it cross-attends them with Z to generate k embeddings $(Y_i)_{i=1}^K$ for the k components in the output mixture of Gaussians.

Given the embedding Y_i for a particular component of the mixture, the mixture likelihood is estimated with a linear projection layer that produces the unnormalized log-likelihood for the component. To generate the trajectory, Y_i is projected using another linear layer to output 4 time series: $T_i = \{\mu_t^x, \mu_t^y, \log\sigma_t^x, \log\sigma_t^y\}_{t=1}^T$, corresponding to the means and log-standard deviations of the predicted Gaussian at each timestep.

For training, the loss is decomposed into separate classification and regression losses. Given k predicted Gaussians $(T_i)_{i=1}^k$, let $\hat{i}$ denote the index of the Gaussian with the mean closest to the ground truth trajectory G. Nayakanti et al. (2022) train the mixture likelihoods on the log-likelihood of selecting the index $\hat{i}$, and the Gaussian $T_{\hat{i}}$ to maximize the log-probability of the ground truth trajectory:

$$\max \log \Pr(\hat{i} \mid Y) + \log \Pr(G \mid T_{\hat{i}}).$$

2.1.3 Motion Forecasting Evaluation Metrics

Following the definitions in Nayakanti et al. (2022), we use four metrics in all our experiments: **brier-minFDE**, **minFDE**, **minADE**, and **Miss Rate (MR)** to compare models. For all metrics, we consider only the top $k = 6$ most likely modes and use only the mean of each mode.

$minDE_k^t$ (Minimum Distance Error): Considers the top-k most likely trajectories output by the model, and computes the minimum distance to the ground truth trajectory at timestep t.

MR^t (Miss Rate): For each predicted trajectory, we compute whether it is sufficiently close (within a threshold of 2 meters) to the predicted agent's ground truth trajectory at time t. Miss rate is the proportion of predicted agents for which none of the predicted trajectories are sufficiently close to the ground truth.

$minADE_k$ (Minimum Average Distance Error): Similar to $minDE_k^t$, but the distance is calculated as an average over all timesteps.

$minFDE$ (Minimum Final Displacement Error): The L2 distance between the endpoint of the

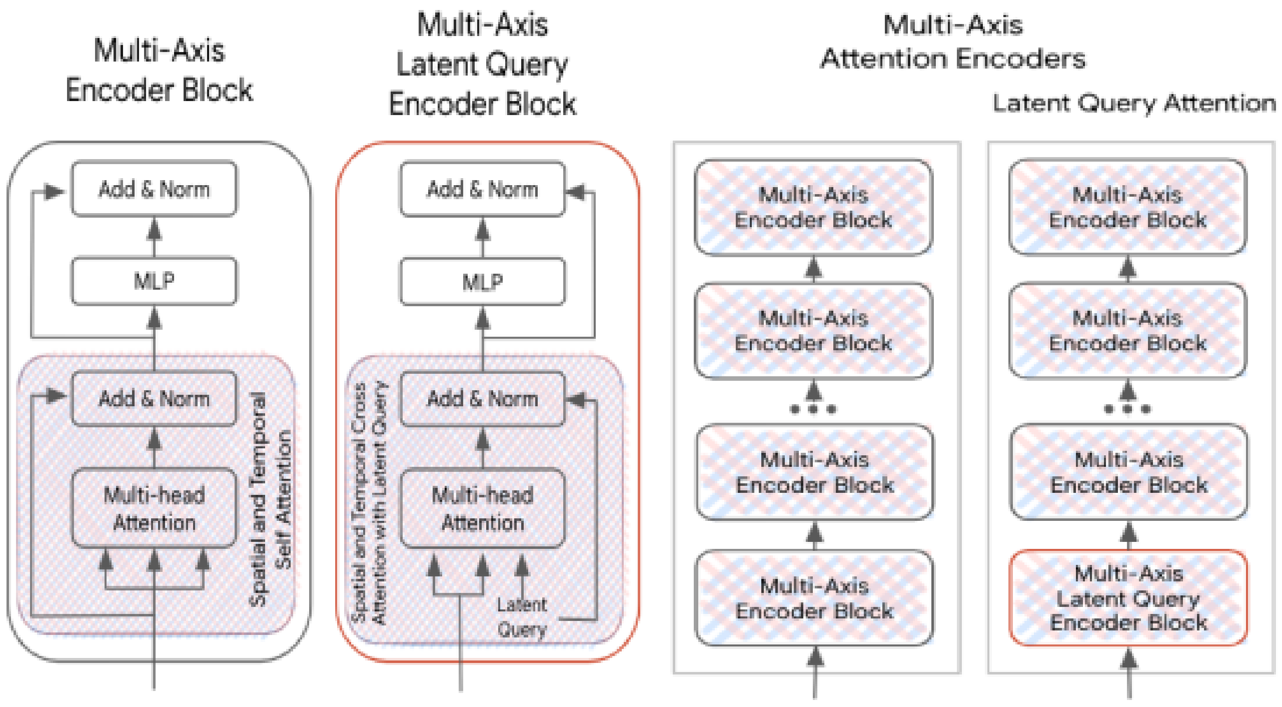

Multi-Axis Encoder Block
Add & Norm
MLP
Add & Norm
Multi-head Attention
Spatial and Temporal Self Attention
Multi-Axis Latent Query Encoder Block
Add & Norm
MLP
Add & Norm
Multi-head Attention
Spatial and Temporal Cross Attention with Latent Query
Latent Query
Multi-Axis Attention Encoders
Latent Query Attention
Multi-Axis Encoder Block
Multi-Axis Latent Query Encoder Block

Figure 4: Wayformer time and spatial attention encoder with latent query

best forecasted trajectory and the ground truth.

brier-minFDE: This is defined as the sum of $minFDE$ and the Brier score $(1-p)^2$, where p is the probability of the best-predicted trajectory.

2.2 Wayformer in Practice: The UniTraj Framework

There is no open-source code available for Nayakanti et al. (2022), but we utilized the implementation provided in **UniTraj** Feng et al. (2024). UniTraj is a comprehensive framework that unifies various datasets, models, and evaluation criteria, presenting new opportunities for the vehicle trajectory prediction field, and it includes Wayformer. Our initial objective was to reproduce the results of Wayformer in UniTraj. The goal is to forecast 6 seconds of future trajectory given 2 seconds of past trajectory data as input.

2.2.1 Dataset, Input Representation, and Encoding Strategy

We worked with nuScenes[5], a public large-scale dataset for autonomous driving included in UniTraj. We chose nuScenes for its smaller size compared to the Waymo Open Motion Dataset[6] used in Nayakanti et al. (2022) and given our computational resources; experiments on the nuScenes dataset with Wayformer are available in UniTraj and served as our baseline. The nuScenes dataset does not provide traffic light signal data, so we only used the other three modalities as inputs.

[5]we present further the dataset in the appendix section A
[6]https://waymo.com/open/

Agent of Interest Past Trajectories: A tensor of shape $[A, T, 1, D]$ is reshaped to $[A, T \times 1, D]$.

Context Agent Trajectories: A tensor of shape $[A, T, S, D]$ is reshaped to $[A, T \times S, D]$.

Road Graphs: A tensor of shape $[A, 1, S_r, D]$ is reshaped to $[A, 1 \times (S_r = N_r \times N_{pr}), D]$ where N_r is the number of road lines (or polylines) and N_{pr} is the number of points per road.

Following the results in Nayakanti et al. (2022), we chose an **early fusion** encoder, so the three inputs are concatenated into a single tensor X_{base} of shape $[A, N = (T \times S + N_r \times N_{pr}), D]$ where N is the total number of tokens. We used latent queries in the first encoder layer to reduce the cost of the attention mechanism in the encoder. The network outputs a tensor of latent features of shape $[A, N_{out}, D]$.

2.2.2 Decoding Strategy

As described in section 2.1.2, the Wayformer decoder in UniTraj uses k learned initial queries $(S_i \in \mathcal{R}^h)_{i=1}^k, S_i \in \mathcal{R}^D$ to cross-attend to the encoder output and generate k embeddings $(Y_i)_{i=1}^K, Y_i \in \mathcal{R}^D$ for the k components in the output mixture of Gaussians, also known as the motion modes. These motion modes are used to feed the two heads of motion forecasting: the trajectory choice head and the trajectory prediction head.

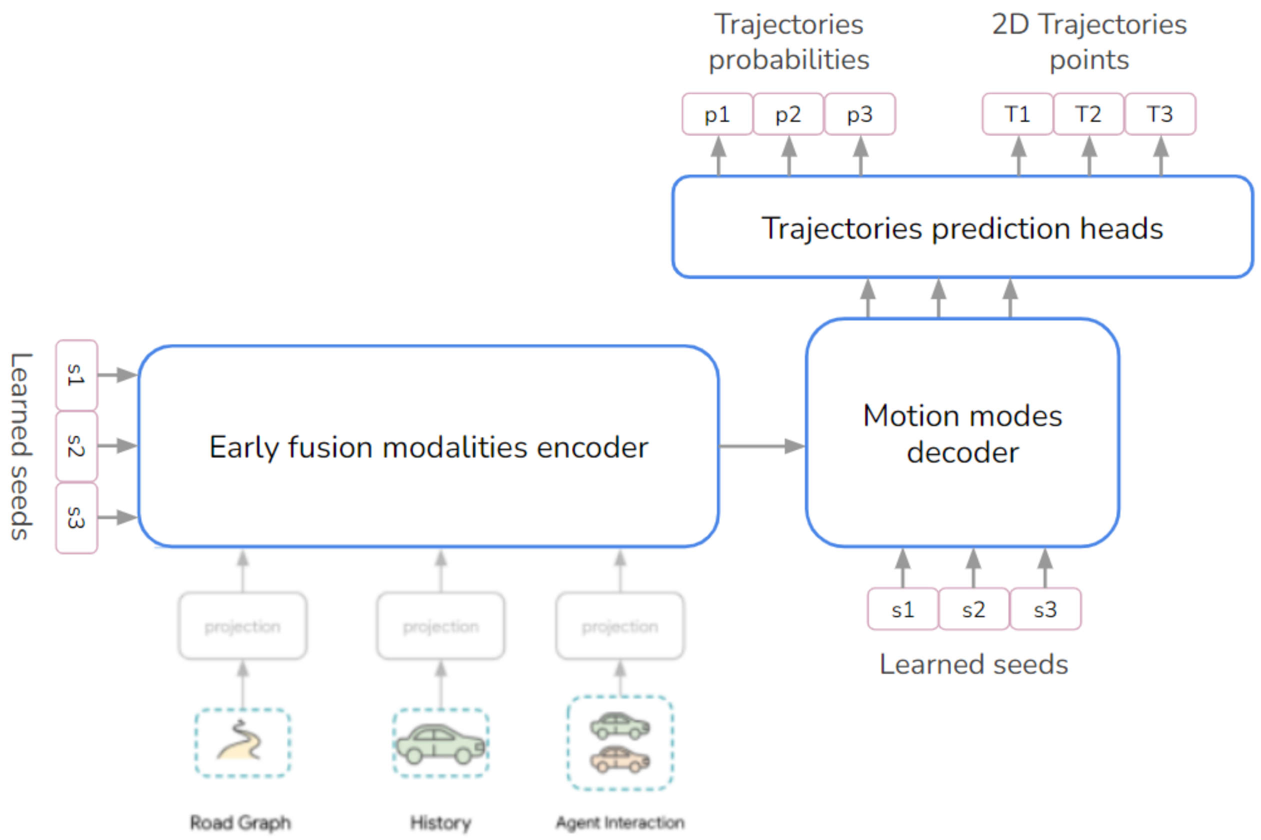


Figure 5: Wayformer network in the UniTraj Framework

2.2.3 Experiments: Baseline Reproduction

In the baseline configuration, the data are sampled at a frequency of 10 Hertz, so we have $T = 20$. The number of context agents for each agent of interest is set to $S = 32$, N_r is set to 384, and N_{pr} to 30. The map range is $(-100, 100)$ meters on the x and y axes around the agent of interest. Note that for each trajectory to forecast, all positions are relative to the agent of interest, with the agent of interest's position centralized to the origin $(0, 0)$.

The total number of input tokens is 12192 for $L_{out} = 192$ latent queries (reduction scale 0.98) in the first encoder layer, and the number of motion modes (or mixture components) k is 6. The encoder have 2 attention layers while the decoder have 8 attention layers. The features dimension was set to $D = 256$.

Following the default UniTraj training configuration, we trained the model using the AdamW optimizer Loshchilov and Hutter (2019) for 150 epochs using a OneCycleLR[7] scheduler, setting the maximum learning rate to 2.10^{-4}, with the learning rate ramping up during the first 2% of training and then gradually decreasing. The scheduler was configured with a *div factor* of 100 and a *final div factor* of 10. We used a batch size of **128** (**256** for the baseline) on **4 Nvidia GeForce RTX 2080 GPUs**. The reproduced metrics compared to those reported in Feng et al. (2024) are shown in Table 1.

2.3 Occupancy and Wayformer

As seen with Wayformer, motion forecasting models can take multi-modal data as inputs. These inputs generally come from a perception module that includes tasks such as agent detection and tracking, as well as road mapping, and must be properly processed to be usable by these models. Motion forecasting datasets like nuScenes provide the curated ground truth data used to train these models. It is important to note that the use of ground truth past trajectories is an inherent assumption of the task, and we are not performing end-to-end motion forecasting (which is beyond the scope of this internship). We assume in our case that these inputs are available, and we do not perform perception tasks.

Our objective is to explore the potential of unifying the different inputs of a motion forecasting model like Wayformer into a single object that can describe the scene effectively, while being more compact, requiring fewer processing steps, and less annotation effort. An ideal candidate for this is the 3D Occupancy, described a point cloud, which is available at a given frequency in motion forecasting datasets.

A point cloud is a set of 3D points (x, y, z), covering a certain range for each of the three axes $(x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max})$, describing the shape of objects present in the environment, provided by a LiDAR sensor. For each point, there is also an intensity value between 0 and 255; certain autonomous driving datasets provide also a semantic label for each point, representing the type of object the point belongs to in the environment (see Figure 6).

A LiDAR point cloud is an excellent descriptor of a 3D scene, showing the geometric shapes of objects, their positions, and even the movement of objects when considering a sequence of point clouds captured at a certain frequency over a time interval. nuScenes provides the LiDAR point cloud at a

[7]https://pytorch.org/docs/stable/generated/torch.optim.lr$_s$*cheduler.OneCycleLR.html*

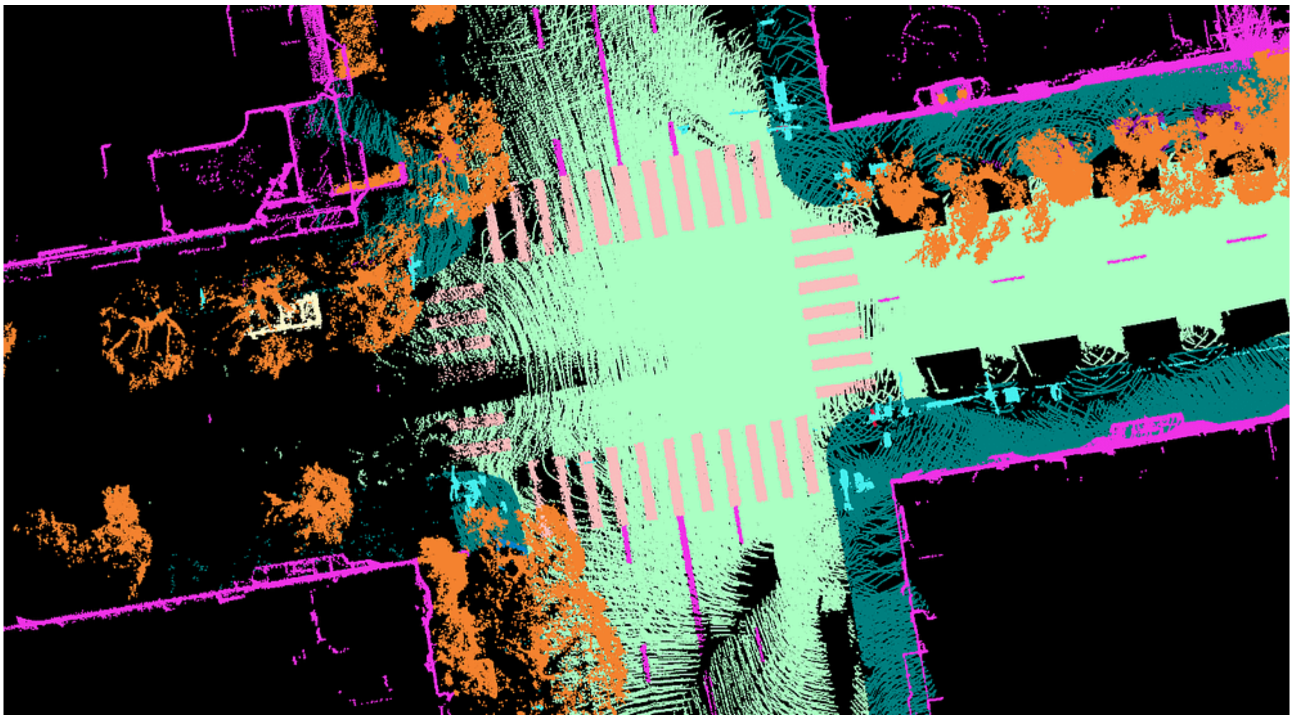


Figure 6: An example of a point cloud with semantic label colors.

sample rate of 2 Hz (2 point clouds per second) for all its driving scenes. The question that arises is: **how can we encode the rich information contained in this sequence of point clouds in an occupancy features map which could be integrated in a transformer-based motion forecasting model like Wayformer**?

This question raises two areas of study: first, we need to encode the point cloud into a latent space with features that adequately describe the scene occupancy; second, we need to find a way to decode this occupancy features map for motion forecasting.

Given that we are working with a transformer-based network, we need to tokenize the occupancy features map to leverage the attention mechanism. If we assume that for each agent of interest, we have a tensor X_{pc} of tokens with shape $[N_{st}, D]$, where N_{st} is the number of tokens encoding the scene, and D is the feature dimension of the tokens, then we can easily include these tokens in a transformer network. Note that X_{pc} is a tensor derived from a point cloud encoder, and its features can contain information about the occupancy of the driving scene, coarse-grained and fine-grained geometric information about the structure of the scene, as well as flow and motion information if we add a time dimension through the sequence of point clouds for T past timestamps. In the following two sections, we will explain how we encoded the point cloud into a tokenized occupancy features map and detail our decoding approaches for motion forecasting using the encoded tensor.

2.3.1 Point Cloud Encoding

We use the architecture of PaSCo Cao et al. (2024) for point cloud encoding. PaSCo, which stands for Panoptic Scene Completion (PSC), was designed to predict multiple variations of PSC given an incomplete 3D point cloud, while allowing uncertainty estimation through mask ensembling. We only use the encoder part of the network, which is mainly composed of a point encoder (1), a Minkowski convolution Choy et al. (2019) network (2), and a dense convolutional network (3).

(1): The point encoder takes as input a point cloud $PC = (x_i, y_i, z_i, I_i)_{1 \leq i \leq N_p}$ with N_p points, each represented by its 3D coordinates and the intensity value given by the LiDAR sensor. It begins by normalizing the input features using batch normalization and then passes them through a series of **3** fully connected layers, which progressively increase the feature dimensionality, with ReLU activations applied after each layer. The final layer projects the features to the desired output dimension. The point coordinates are then voxelized to create a 3D volume before proceeding to step (2). We denote H, W, and L as the number of voxels along the x, y, and z dimensions of the scene, respectively.

(2): The Minkowski network is a 3D convolutional neural network designed to process sparse tensor inputs with varying feature maps at multiple levels. It takes the voxelized 3D coordinates from the point cloud and the features from the point encoder, and starts by applying a Minkowski Convolution to the input features, followed by a series of **3** hierarchical encoding blocks that progressively downsample the spatial dimensions by a factor of **2** while increasing the feature depth by a factor of **2**. Each block consists of sparse convolutions, combined with batch normalization, ReLU activation, and dropout layers. The output this step is a compact and dense high-level 3D voxels volume of dimensions $[H/8, W/8, L/8]$, each voxel representing a region of the scene described by a D-dimensional features vector.

(3): The dense encoder is a ResNet-18 He et al. (2015) model that further processes the output of step (2), with the z-dimension treated as a feature dimension. This choice was motivated by the goal of using 2D convolutions, which are much more efficient and as effective as 3D convolutions, as shown in Yang et al. (2020). The network applies 2D convolutions with padding to maintain the spatial resolution of $[H/8, W/8]$ along the x and y dimensions, outputting feature maps with D channels. The final output is a 2D occupancy feature map of shape $[H/8, W/8]$.

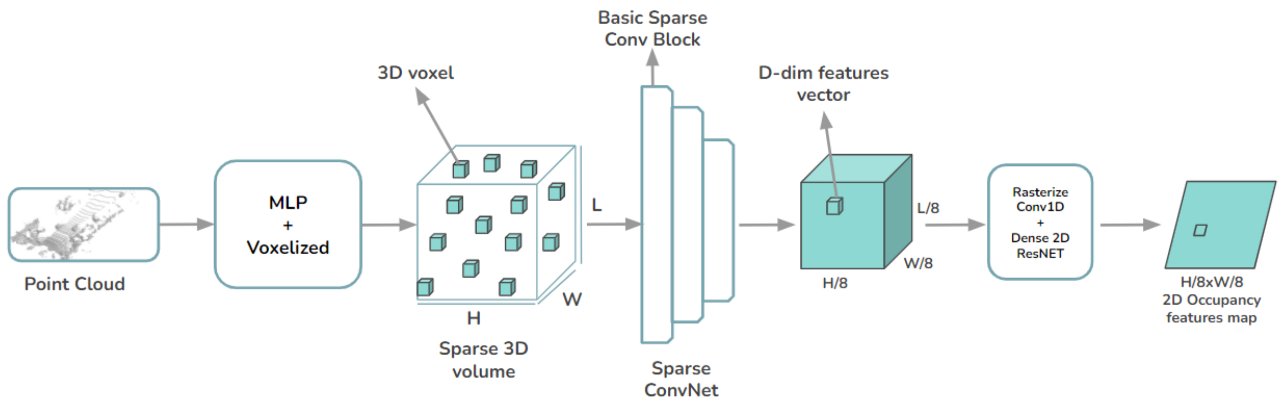


Figure 7: Our Point Cloud Encoder Architecture

2.3.2 Decoding Strategy for Occupancy in Motion Forecasting

We explored two main approaches to incorporate occupancy information into Wayformer. The first, more intuitive and straightforward, involves adding the tensor of tokens output by the point cloud encoder as additional inputs in the Wayformer encoder through concatenation with the existing input tokens. The entire network, including the point cloud encoder, is then trained end-to-end using the motion forecasting loss. Although one might assume that the latent query attention in the first layer of the encoder could simply ignore these new tokens from the point cloud encoder—since the motion forecasting loss might not directly incentivize training such an encoder—this basic approach revealed interesting insights that guided the rest of our research.

(1) Specifically, in this first approach, we replace the base input tensor of Wayformer in UniTraj, X_{base} (see Section 2.2.1), with $X_{occ} = concat([X_{base}, X_{pc}])$, where concat denotes concatenation along the token dimension, and X_{pc} (see Section 2.3) represents the tokenized encoding of the point cloud. We refer to this method as **early fusion** Way<u>occ</u>former (Figure 8). It is important to note that we only encode the point cloud recorded by the LiDAR sensor at the last past timestamp.

(2) **Late fusion** Way<u>occ</u>former: We developed a more advanced approach to integrate occupancy information into Wayformer, called late fusion Wayoccformer. Unlike the natural intuition in forecasting, which typically involves using past values to predict future ones, Wayformer encodes all available symbolic scene information, including the past trajectories of the agents of interest, into a single latent representation which is decoded to predict the trajectories. Even though this approach leads to state-of-the-art performance Nayakanti et al. (2022), we explored the idea of augmenting the features learned in the motion modes with explicit features representing the agents of interest motion state information enhanced with the scene occupancy information. More specifically, we build a parallel network as followed: given the tensor of agents of interest at the last past timestamp, X_{agi}, with a shape of $[A, 1, D]$, an attention network **AgiOcc** uses X_{agi} as queries to cross-attend with $X_{pc} \in \mathcal{R}^{A \times N_{st} \times D}$ and returns a tensor X_{agiocc} of the same shape as X_{agi}. The features of X_{agiocc}, which contain information about the agents of interest at the last timestamp before forecasting contextualized with occupancy information at the same timestamp, are used to enhance the motion modes learned by the base Wayformer decoder in two different manners, an explicit one **EO** and an implicit one **IO**. Figure 9 illustrates these two strategies.

- **EO**: **E**xplicit **O**ccupancy awareness via cross-attention with all occupancy map tokens—As mentioned in Section 2.3.1 (3), the encoder returns a feature map of shape $[H/8, W/8]$, with each cell containing occupancy information about a region of the scene described by D features. This map needs to be tokenized to be manageable by an attention network. Following a commonly adopted approach in the literature, we flatten the entire map to have a sequence of $H/8 * W/8$ tokens. This gives us $X_{pc} \in \mathcal{R}^{A \times H/8 * W/8 \times D}$, which is used as keys and values, with $X_{agi} \in \mathcal{R}^{A \times 1 \times D}$ as queries. We do not add any positional embeddings to X_{pc} as the position information is already learned by the encoder. Each agent $i, 1 \leq i \leq A$ in the occupancy-aware output features $X_{agiocc} \in \mathcal{R}^{A \times 1 \times D}$ is then repeated k times, with k being the number of motion modes in the Wayformer decoder, and concatenated to the agent's motion mode embeddings $(Y_i)_{i=1}^{K}, Y_i \in \mathcal{R}^{D}$ along the feature dimension. The new motion modes embeddings $(Y_i)_{i=1}^{K}, Y_i \in \mathcal{R}^{2D}$ are used to feed the motion forecasting heads.

– **IO**: **I**mplicit **O**ccupancy awareness via offset learning—Inspired by Agro et al. (2023), which proposes a unified approach that implicitly represents occupancy and flow over time with a single neural network for perception and future prediction in autonomous driving, we explore the potential of using a network to implicitly learn the relevant region cells for agent trajectory forecating in the occupancy feature maps. More specifically, given the occupancy feature map of shape $[H/8, W/8]$, centered on the agent of interest's position at the last timestamp, we collect the features $z_{agi} \in \mathcal{R}^D$ at the agent of interest's 2D location $r_{agi} \in \mathcal{R}^2$. We then predict K reference points $r_1, ..., r_K$ by offsetting $r_k = r_{agi} + \Delta q_k$, where the offsets $\Delta q_k \in \mathcal{R}^2$ are computed by employing a fully connected residual network on z_{agi} and the agent of interest's state feature vector $z_{agiT} \in \mathcal{R}^D$. The intuition behind this is to use the current motion state information of the agent of interest, combined with the current scene occupancy information at his position to implicitly learn its possible future trajectory directions.

  We then collect a set of K feature vectors on the occupancy feature map at the reference points and concatenate them to the motion modes $(Y_i)_{i=1}^K, Y_i \in \mathcal{R}^D$. The new modes $(Y_i)_{i=1}^K, Y_i \in \mathcal{R}^{2D}$ are used to feed the motion forecasting heads.

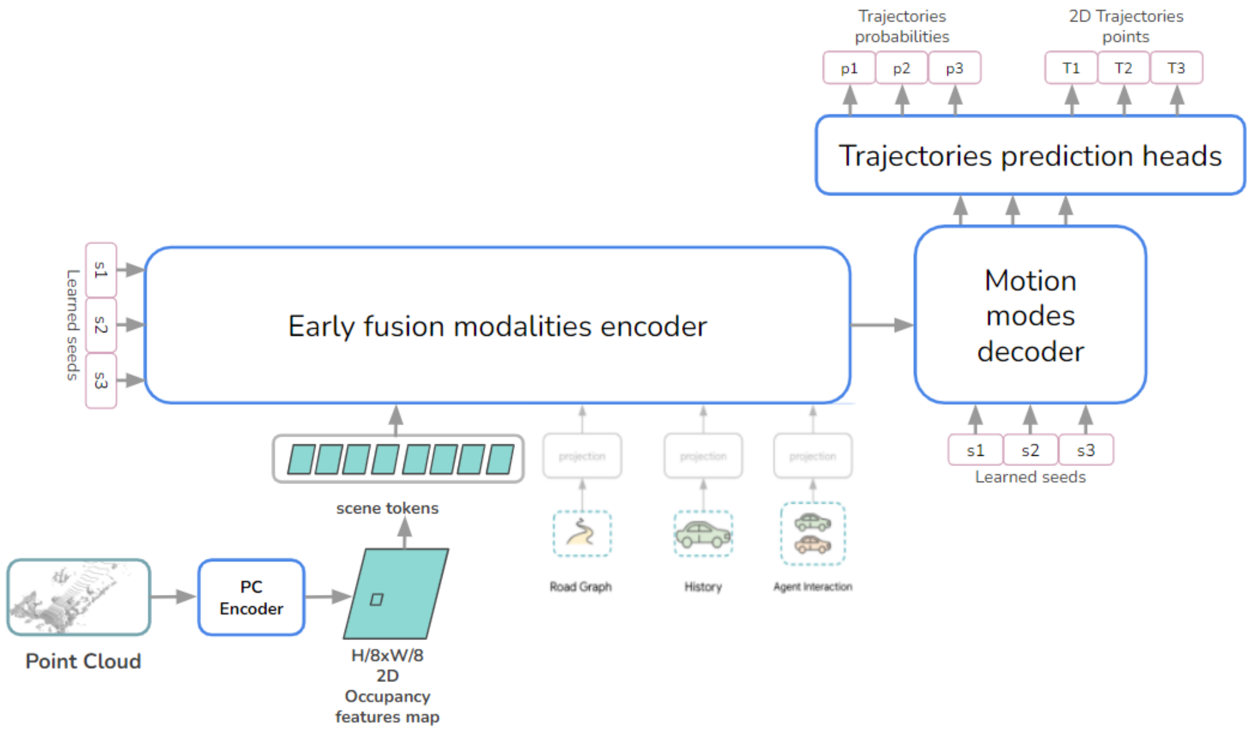


Figure 8: Early fusion Wayoccformer, the point cloud is encoded in an occupancy features map, which is tokenized and added as additional input in the Wayformer early fusion encoder.

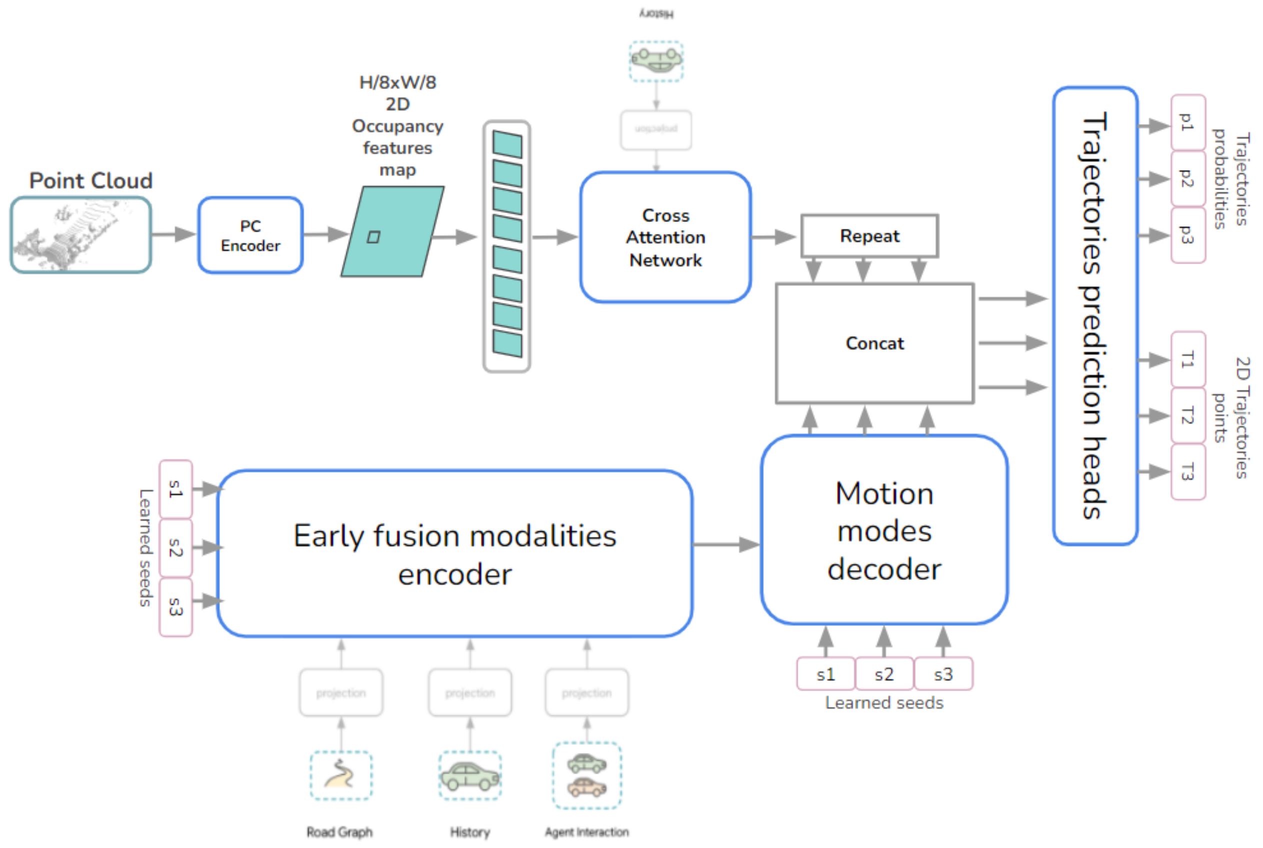


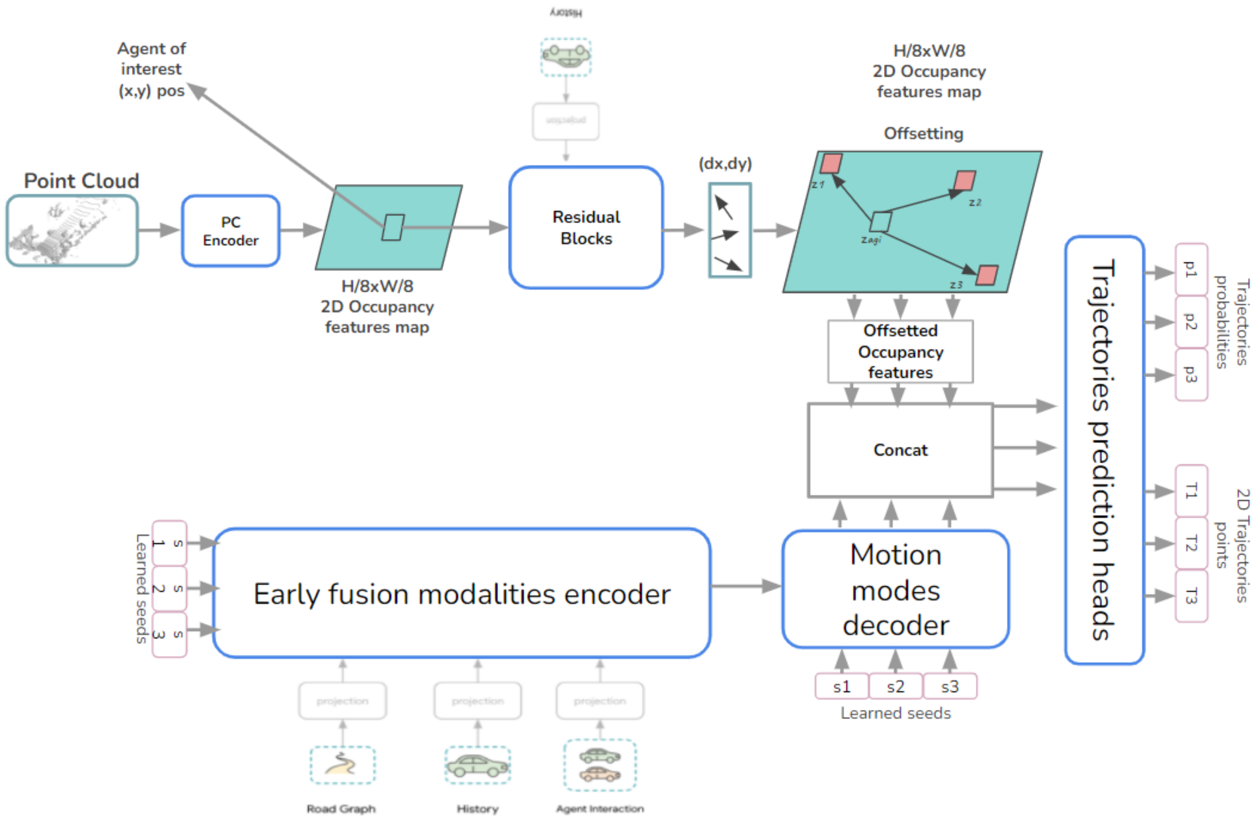
Figure 9: Late fusion Wayoccformer with explicit occupancy attention (EO), we build a parallel network that leverage agent history state information to cross attend the occupancy tokens from the scene encoder; the outputed agent occupancy aware features are then concatenated with the Wayformer decoder motion modes to predict the futur trajectories.

2.3.3 Experiments and Results

We trained and evaluated all the Wayoccformer networks (early fusion, late fusion: EO, and IO) on the nuScenes dataset in an end-to-end manner from scratch with the basic motion forecasting loss. We used the same training parameters as in section 2.2.3, except that we decreased the batch size to 32 for computational reasons, as the point cloud encoder demands significant memory, and we increased the number of epochs from 150 to 250 to allow the (larger) network more time to train. The three trainings were done on **4 NVIDIA GeForce RTX 2080 GPUs** and it's took 2 days and 3 hours, 2 days and 13 hours and 2 and days 1 hour respectively for the early fusion, late fusion EO and late fusion IO versions to fully converge.

Table 1 compares the scores reported by UniTraj Feng et al. (2024) with our reproduced results on the nuScenes dataset, for Wayformer and Way**occ**former (Wayformer with occupancy). The global alignment between the reproduced scores and the original results across all the metrics—Brier-minFDE, minFDE, minADE, and MR—demonstrates that we were able to effectively replicate the Wayformer's training in UniTraj, validating the reliability and consistency of our experimental setup.

For Way<u>occ</u>former, in the context of the **early fusion** approach, we observe a slight increase in



Agent of interest (x,y) pos
History
Point Cloud
PC Encoder
H/8xW/8 2D Occupancy features map
Residual Blocks
(dx,dy)
H/8xW/8 2D Occupancy features map
Offsetting
Offsetted Occupancy features
Concat
Trajectories prediction heads
p1
p2
p3
Trajectories probabilities
T1
T2
T3
2D Trajectories points
Learned seeds
Early fusion modalities encoder
Motion modes decoder
s1
s2
s3
Learned seeds
Road Graph
History
Agent Interaction

Figure 10: Late fusion Wayoccformer with implicit occupancy attention (IO), we use the occupancy features at the position of the agent of interest and the agent history state information to predict K offsets through a residual network; the features at the offsets position are then concatenated with the Wayformer decoder motion modes to predict the futur trajectories.

the Brier-minFDE and minADE metrics compared to the reproduced Wayformer results. Specifically, the Brier-minFDE metric increases from 3.071 to 3.098, and minADE increases from 1.041 to 1.118. This indicates that the early fusion strategy, while effectively integrating the occupancy information, might have introduced some noise or less effective utilization of the occupancy features, leading to a marginal drop in predictive accuracy. However, it is noteworthy that the Miss Rate (MR) improves significantly from **0.413** to **0.3713**, suggesting that early fusion helps the model make more confident predictions, reducing the frequency of large forecasting errors in the most of scenarios in the dataset. This improvement in MR also indicates that the model is better at correctly predicting the trajectory modes that matter the most, even if the average distance error (minADE) increases slightly.

The **late fusion** approaches, particularly with the **EO** (Explicit Occupancy) strategy, show notable improvements in all metrics. The Brier-minFDE decreases to **2.944** (-0.116), and minFDE reaches **2.291** (-0.209) compared to the UniTraj reported scores. The minADE also improves slightly to **1.008**, and the MR (Miss Rate) is reduced to **0.366** (-0.054%). These results prove the late fusion EO strategy is more effective in leveraging the occupancy information, leading to clearly better overall performance.

The **late fusion IO** (Implicit Occupancy) strategy also performs well, with a Brier-minFDE of

Metrics	Brier-minFDE	minFDE	minADE	MR
Wayformer				
UniTraj Feng et al. (2024)	3.06	2.50	1.04	0.42
Reproduced (ours)	3.071	2.432	1.041	0.413
Way**occ**former				
Early fusion (ours)	3.098	2.434	1.118	0.3713
Late fusion (EO) (ours)	**2.944**	**2.291**	**1.008**	**0.366**
Late fusion (IO) (ours)	2.952	2.315	**1.008**	0.3982

Table 1: Wayformer reproduced scores and Wayoccformer scores with the early fusion strategy and late fusion strategies (EO and IO). While the early fusion method struggled to match the performance of the baseline in all metrics except for the miss rate, our late fusion network especially with the EO appraoch largely outperforms Wayformer in all metrics.

2.952 (−0.116) and a minFDE of **2.315** (−0.185). While these scores are slightly higher than the EO variant, the minADE remains the same at **1.008**. The MR, however, increases to **0.3982** possibly due to the complexity of the implicit learning process not fully capturing all relevant occupancy information for the trajectories prediction.

3 Related Work

In this section, we review recent advancements in motion forecasting, with a focus on approaches that integrate raw sensor data, such as LiDAR, to enhance prediction accuracy. We first discuss WOMD-LiDAR Chen et al. (2024), which addresses limitations in traditional motion forecasting models by incorporating fine-grained LiDAR data into the forecasting pipeline. We then explore MoST Mu et al. (2024), which extends this approach by combining LiDAR and image data through advanced multi-modal tokenization techniques. Finally, we highlight how our work diverges from these methods by focusing on end-to-end training with occupancy-centric features, refining trajectory predictions through occupancy-aware scene decoding.

3.1 WOMD-LiDAR: A Raw Sensor Dataset Benchmark for Motion Forecasting

The motivation behind the paper Chen et al. (2024) is to address a significant challenge in motion forecasting: the heavy reliance on abstract representations of the environment, which often fail to capture the complexity and richness of real-world scenes. Traditional motion forecasting models, like Wayformer (Section 2.1.1), typically use high-level features such as 3D bounding boxes and polylines to represent objects and road geometries. However, these representations, derived from perception models, are inherently lossy and may omit critical fine-grained scene information. This limitation motivates the integration of raw sensor data, such as LiDAR, into motion forecasting models, aiming to provide a more detailed and accurate representation of the environment.

The paper introduces the **WOMD-LiDAR** dataset as a significant step toward addressing these challenges. This dataset augments the Waymo Open Motion Dataset (WOMD) with high-quality Li-

DAR point clouds, offering a more comprehensive and fine-grained view of driving scenes. The authors propose a two-stage approach for motion forecasting: first, a perception model extracts embedding features from the raw LiDAR data, and then these embeddings are fed into a motion forecasting model—specifically the Wayformer model—in an early fusion manner. By doing so, they aim to enhance the accuracy of motion predictions by leveraging the rich information contained in the raw sensor data.

The LiDAR encoding scheme employed in the WOMD-LiDAR approach is fundamentally object-centric, focusing on extracting and utilizing rich features related to objects detected in the environment. The method begins by leveraging a pre-trained SWFormer Sun et al. (2022) model, initially trained for 3D object detection tasks. The embedding features, which are used to produce detection results in the detection heads, are extracted as input to the scene encoder of the Wayformer model. The output tensor E from SWFormer is an $N \times T \times C$ tensor, where N is the number of detected boxes, T is the number of input frames per scene, and C is the feature size. To adapt E to be compatible with the input requirements of the Wayformer scene encoder, the first two dimensions are flattened into the token dimension, and a one-layer Axial Transformer Ho et al. (2019) is applied as a LiDAR encoder to project the output tensor E into a fixed M-token tensor $E \in \mathcal{R}^{M \times C}$, with the same feature size as other modalities (Figure 11).

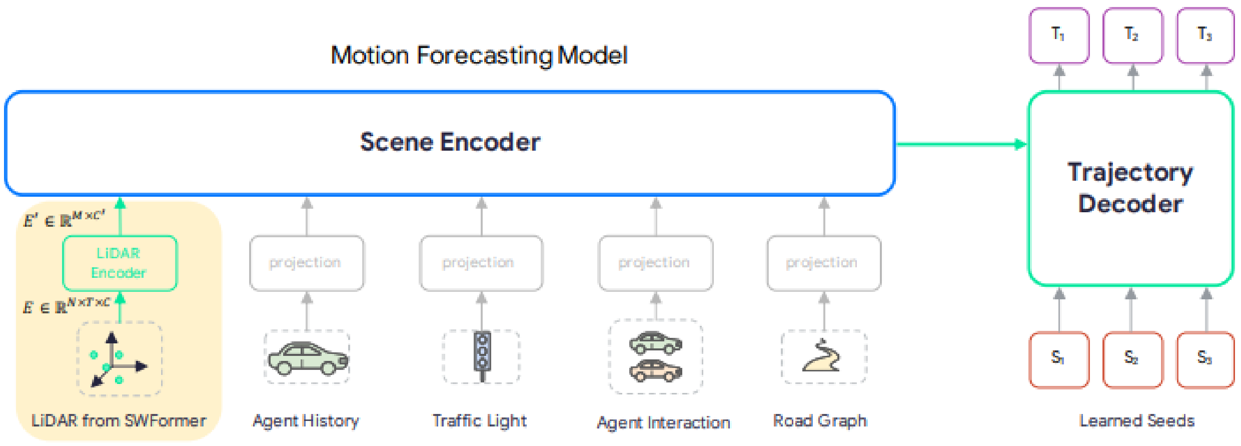


Figure 11: WOMD-LiDAR encoding and Wayformer

3.2 MoST: Multi-modality Scene Tokenization for Motion Forecasting

Similar to WOMD-LiDAR, the paper Mu et al. (2024) aims to extract detailed fine-grained and coarse-grained scene information to augment the symbolic perception inputs of the Wayformer scene encoder. The authors propose an advanced pipeline to extract multi-modal scene tokens from images and LiDAR point clouds, leveraging pre-trained image foundation models and LiDAR points clustering for scene element detection. Figure 12 provides an overview of their scene tokenization pipeline.

The method takes as input multi-view camera images and a full scene point cloud. A pre-trained image foundation model is used to obtain descriptive feature maps, and the scene is decomposed into disjoint elements via clustering. Based on the sensor calibration information between the camera and LiDAR, point-wise image features are obtained. From scene decomposition, each point is assigned a token or cluster ID, and box information is derived for each element. Finally, a feature embedding is extracted for each scene element.

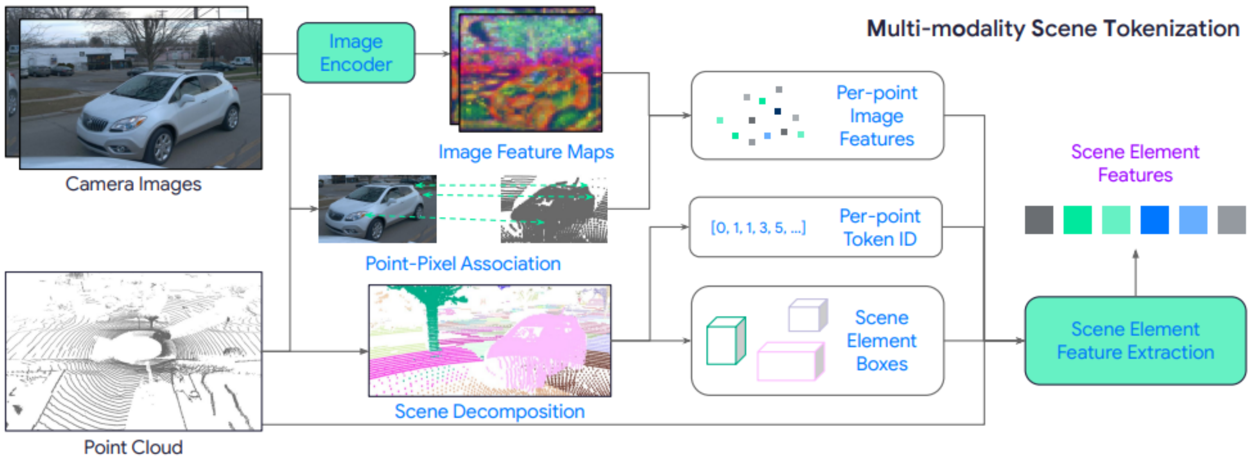


Figure 12: MoST encoding pipeline

The symbolic perception output (Section 2.1.1) and the multi-modality scene tokens are then fused in an early fusion manner to enhance the scene encoding (Figure 13).

3.3 Limitations Compared to Our Work

While WOMD-LiDAR introduces valuable enhancements by incorporating LiDAR data into motion forecasting, and MoST goes further by proposing advanced multi-modal scene tokenization from images and LiDAR point clouds, our approach diverges in two crucial ways:

- **Occupancy-Centric Encoding and End-to-End Training**: WOMD-LiDAR and MoST employ an object-centric approach, with the former relying on a pre-trained object detection network to extract features from raw LiDAR data, and the latter on pre-trained image foundation models combined with LiDAR point clustering for scene element detection. These extracted features are then added as additional inputs for scene encoding in the base motion forecasting model, maintaining a separation between perception and prediction tasks. However, this two-stage pipeline can create dependencies where the quality of motion forecasts is constrained by the performance of the pretrained perception model. In contrast, our approach directly targets occupancy features by adopting a scene-centric encoding strategy. We integrate the LiDAR encoder into the motion forecasting model to leverage occupancy features and train the entire network end-to-end solely with the motion forecasting loss. This holistic integration allows us to evaluate the capacity of the motion forecasting task to train an occupancy network and to explore the utility of the learned occupancy features more effectively, capturing the full complexity of the driving environment without the bottlenecks of a separate perception stage.

- **Refining Trajectory Modes with Advanced Agent Occupancy-Aware features**: While WOMD-LiDAR enhances motion forecasting by integrating LiDAR data, it remains primarily focused on improving existing models with object-centric features. Our method goes further by exploring more sophisticated ways to incorporate occupancy information into the motion forecasting pipeline. Specifically, we introduce a late fusion network that leverages agents' past

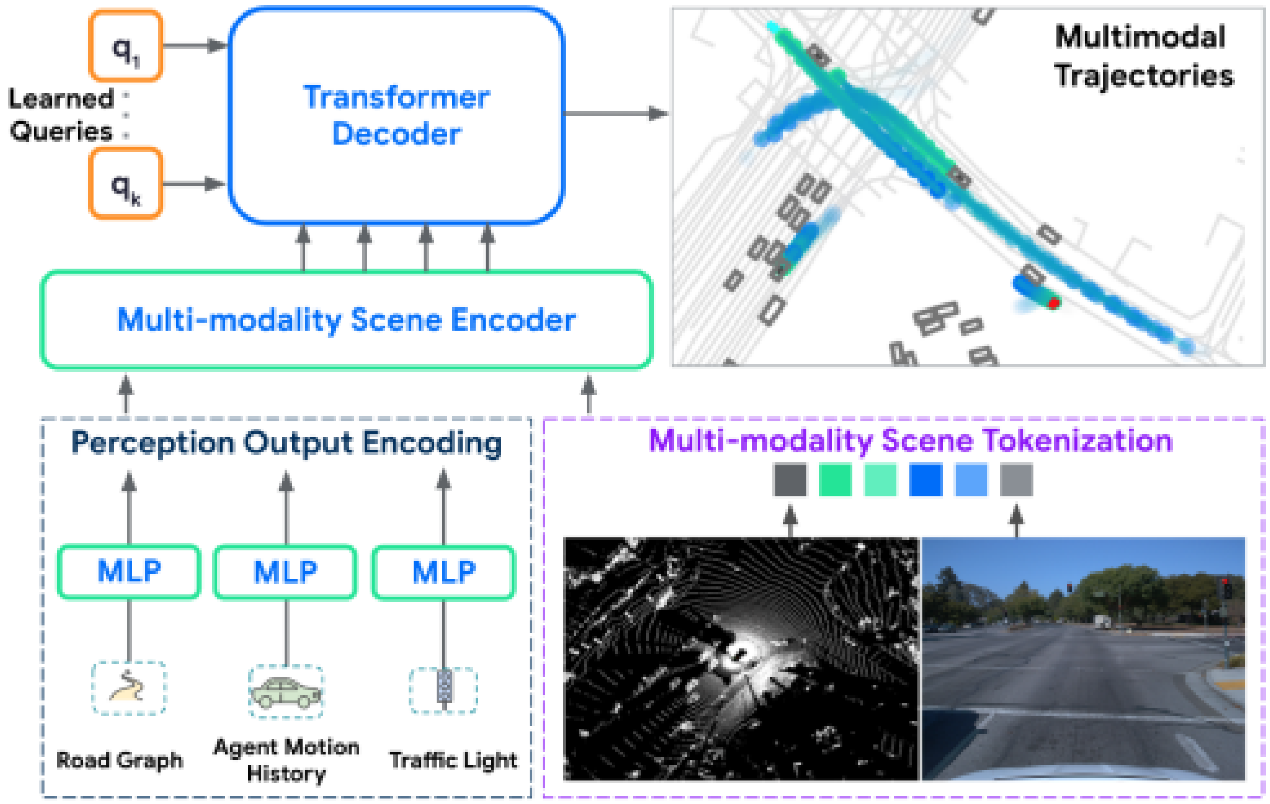


Figure 13: Overview of MoST image and LiDAR encoding pipeline

trajectories as queries for explicit or implicit scene decoding. This design is inspired by natural driving behavior, where drivers use their prior knowledge of the road and surroundings to gather global information about other agents. Before making decisions, they often take a quick global look at the environment (explicit) or focus on key points (implicit) to adapt their trajectory. This step is a crucial missing component in traditional encoding-decoding approaches, and our network aims to address this by providing a more accurate and contextually aware motion forecasting model that better mimics real-world driving behavior.

4 Wayoccformer Ablation Study: Towards Occupancy-Based Motion Forecasting.

As mentioned in Section 2.3, motion forecasting models use ground truth symbolic perception outputs that require significant effort in annotation and processing to be usable by these models. One of these inputs, the context agents tensor, which contains state information about the N closest agents to the agent of interest, is particularly challenging to obtain. It requires precise knowledge of all agents' positions in the scene at each past timestamp, with the feature vectors of each agent at each moment serving as tokens for scene encoding.

An interesting question arises: given that we have a scene encoder, could we replace the well-curated tensor containing the ground truth state information about the context agents, of shape $[A, T*S, D]$, with the tensor of shape $[A, H/8*W/8, D]$ from the point cloud encoder, where each cell (viewed as a token here) represents a region of the scene? In this setup, we definitely need to include the point cloud at multiple past timestamps to capture the scene dynamics. To incorporate the dynamics through multiple point cloud timestamps, we encode each of the 4 past timestamp (2 seconds) point clouds with our point cloud encoder, obtaining a set of 4 encoding tensors of shape $([A, H/8*W/8, D])_{t,1\leq t\leq 4}$. We then concatenate them along the feature dimension D and fuse them using 1D convolution. This results in a new tensor ST of shape $[A, H/8*W/8, D]$ which contains not only the scene information at the last timestamp before forecasting but also dynamic information about the scene motion and flow over the past 2 seconds. We use this tensor in two ways presented in figures 14 and 15.

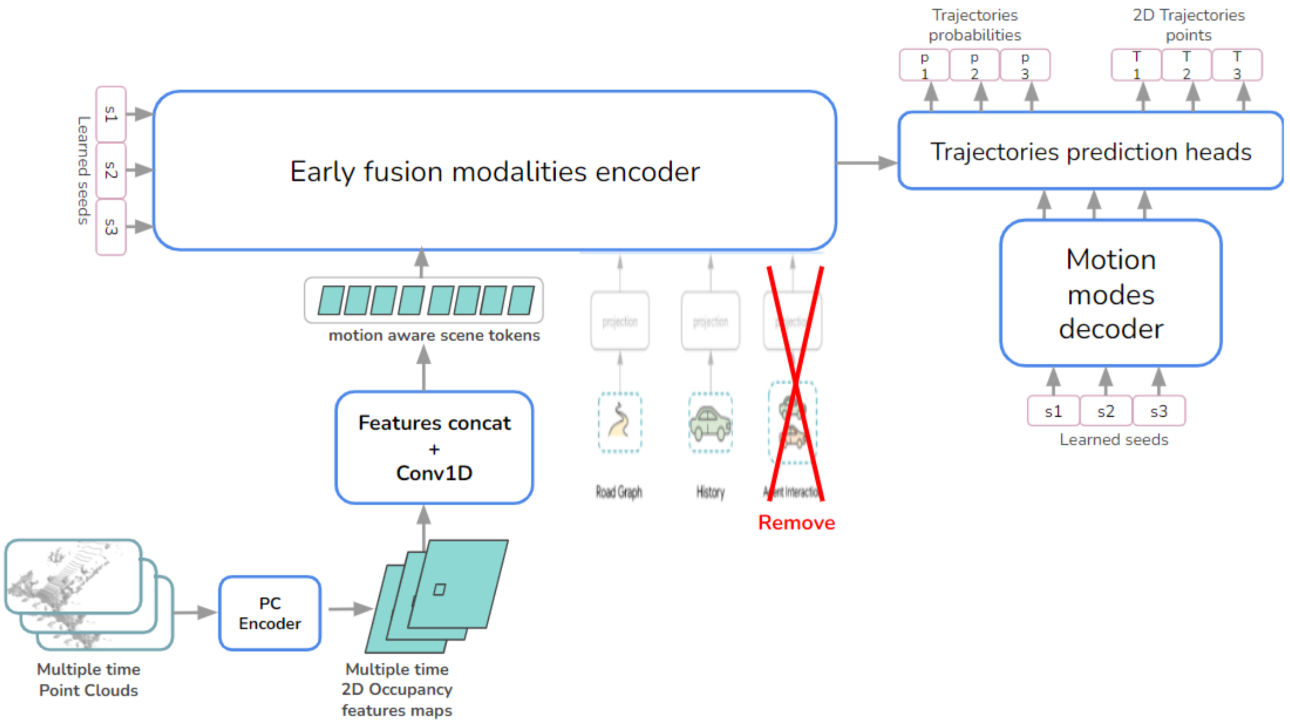

Trajectories probabilities
p 1
p 2
p 3
2D Trajectories points
T 1
T 2
T 3
Learned seeds
s1
s2
s3
Early fusion modalities encoder
Trajectories prediction heads
Motion modes decoder
motion aware scene tokens
projection
Features concat + Conv1D
Road Graph
History
Agent Interaction
Remove
Multiple time Point Clouds
PC Encoder
Multiple time 2D Occupancy features maps

Figure 14: Context Agents Ablation: Early Fusion

- First, we remove the context agent tensor in Wayformer and replace it with ST without changing anything else in the architecture.
- Second, we remove the context agent tensor in Wayformer, but we do not replace it with ST. Instead, we build a parallel network similar to the EO late fusion approach, using the agent of interest's past trajectories tensor $[A, 1, D]$ to query explicitly the past motion-aware scene encoding tensor ST. We replicate each agent output token K times and concatenate them with the K motion modes $(Y_i)_{1\leq i\leq K}$ for trajectory prediction.

We trained each of the two version of the model (early fusion and late fusion) using the same training parameters as in Section 2.2.3 on **2 Nvidia A100 40Gb GPUs** with a total batch size of 32. As we are encoding multiple timestamps LiDAR point cloud, the training naturally required more

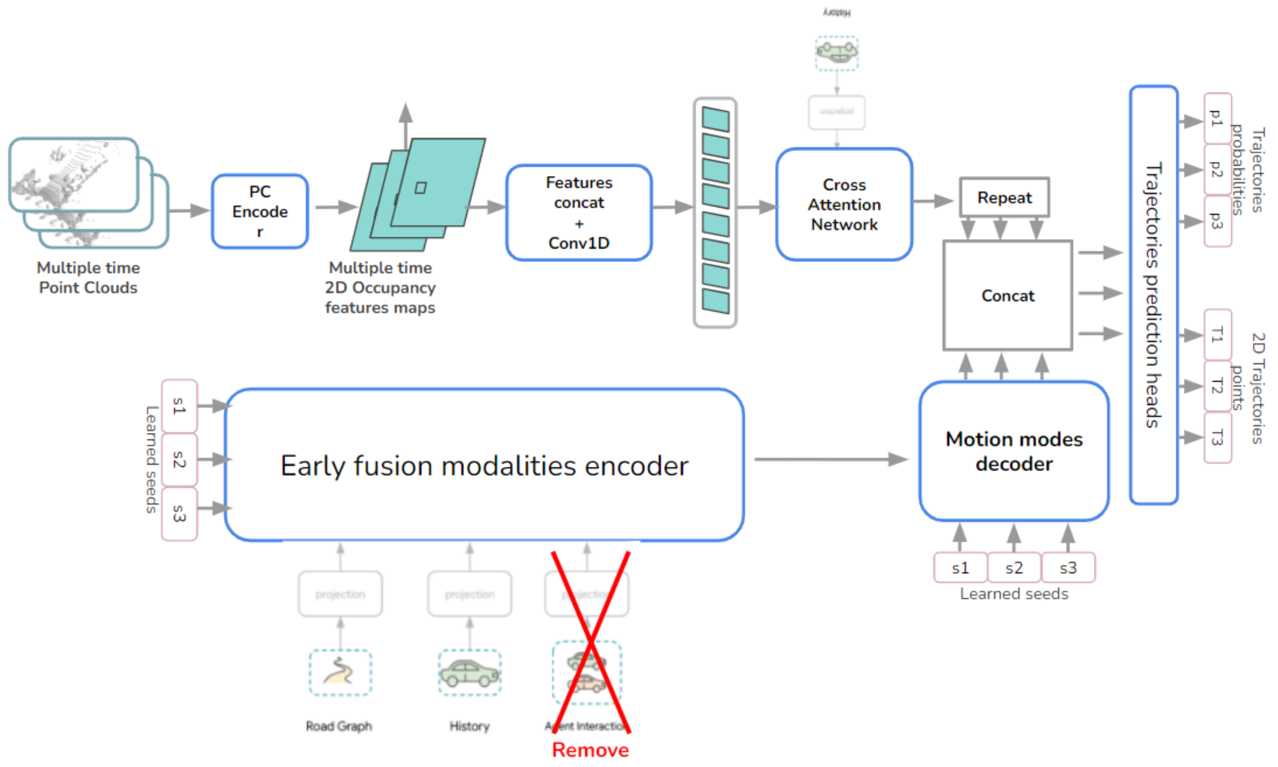


Figure 15: Context Agents Ablation: Late Fusion

compute and it's took almost 3 days for each model to fully converge. The results are presented in Table 2.

As expected, removing the context agents results in a significant drop in performance across all metrics, regardless of the fusion strategy. This is not surprising, as we are replacing well-curated ground truth data with raw sensor data that can be noisy. However, the scores are promising and can be improved with better multiple LiDAR encoding strategies, additional points features (e.g. semantic labels) and more effective fusion strategies. Given the significant advantage of using raw sensor data (much more easier to obtain at a large scale) compared to well-curated ground truth data, we will continue to explore this approach until the end of the internship.

5 Semantic Occupancy for Motion Forecasting

5.1 Adding Semantic Labels in LiDAR Points Features

We studied the impact of incorporating semantic labels as additional features for representing LiDAR points before encoding. In this setup, the point encoder presented in Section 2.3.1 takes as input a point cloud $PC = (x_i, y_i, z_i, I_i, l_1, \ldots, l_s)_{1 \leq i \leq N_p}$, where $(l_1, \ldots, l_s)$ represents a one-hot encoding vector for the semantic label of each point. We utilize the dataset from Occ3D Tian et al. (2023), which provides 3D volume semantic labels for each scene in the nuScenes dataset. Each 3D volume is centered on the ego vehicle and labeled at a resolution of $256 \times 256 \times 16$ (0.4 meter per voxel), covering only a map range of $(-40\text{m}, -40\text{m}, -1\text{m})$ to $(40\text{m}, 40\text{m}, 5.6\text{m})$ along the x, y, and z axes, respectively.

Since the semantic labels are not available for all the points in the full map, we consider only

Metrics	Brier-minFDE	minFDE	minADE	MR
Wayformer				
UniTraj Feng et al. (2024)	3.06	2.50	1.04	0.42
Way**<u>occ</u>**former				
Early Fusion W/O CAG	**3.254**	**2.603**	**1.116**	0.453
Late Fusion W/O CAG	3.299	2.666	1.132	0.496
AutoBot Feng et al. (2024)	3.36	2.62	1.21	**0.40**

Table 2: Wayoccformer without Context Agent (W/O CAG), we observe a drop in performance across all metrics for the two strategy; the early fusion method work better than the late fusion one, highlighting the need to have context scene information earlier in the network. But we notice that on three metrics, the scores are better than Autobot, another MF model included in UniTraj that's leverage curated ground truth context agents.

the points within the aforementioned map range and encode the scene similarly to Section 2.3.1. Although this approach may not capture structural information from areas far from the ego vehicle, we hypothesize that the availability of semantic labels could compensate by providing fine-grained semantic information close to the ego vehicle. This could potentially enhance trajectory forecasting accuracy, even for vehicles farther than 40 meters from the ego vehicle.

We trained the entire network by adding semantic point labels both in an early fusion manner and via explicit late fusion. We did not employ the implicit late fusion method, as it requires positioning the vehicle of interest within the occupancy map—a task not feasible for all agents of interest, given the restricted map range.

To quantify the impact of adding semantic labels, we conducted experiments both with and without the inclusion of these semantic labels, using the same restricted map range for the point cloud. All trainings were done with the same parameters as in Section 2.3.3. The results are presented in Table 3.

Results and Discussion

It is important to **highlight a discrepancy** between the UniTraj baseline scores reported in Table 1 and those in Table 3. The scores in Table 3 (UniTraj Feng et al. (2024) and our reproduced ones) are the original results we obtained at the beginning of the internship and remained consistent for the first three months. At that time, despite using the exact setup as reported in the paper and codebase, and even after multiple discussions with the authors, we were'nt able to reproduce their results. After considerable effort, we decided to use our reproduced scores as the baseline and build upon them for subsequent experiments. As shown in Table 3, we observe that incorporating LiDAR features alone provides noticeable changes compared the reproduced baseline scores.

The early fusion model with and without semantic labels performs worse than the baseline. While simpler, this approach does not optimally utilize the additional information, resulting in a significant drop in performance across the first three metrics.

The opposite occurs with the EO late fusion approach. The model shows a slight improvement

Metrics	Brier-minFDE	minFDE	minADE	MR
Wayformer				
UniTraj Feng et al. (2024)*	3.85	2.83	1.33	0.48
Reproduced (ours)	3.486	2.84	1.329	0.538
Way**occ**former				
Early Fusion W Sem (ours)	3.728	3.066	1.44	0.538
Early Fusion W/O Sem (ours)	3.853	3.188	1.501	0.526
Late Fusion W Sem (EO) (ours)	**3.365**	**2.72**	**1.309**	**0.519**
Late Fusion W/O Sem (EO) (ours)	3.453	2.814	1.328	0.527

Table 3: Wayoccformer with (W Sem) and without (W/O Sem) semantic labels on restricted map, early fusion, and EO late fusion. * indicates a discrepancy between the UniTraj Feng et al. (2024) scores reported here and those in Table 1 due to an update in the codebase and the paper by the authors during the internship.

across the four metrics compared to the reproduced baseline by using the restricted LiDAR only without point semantic labels. With semantic labels, the improvements are more pronounced, with the model outperforming the baseline and its counterpart without semantic labels across all metrics. This highlights the effectiveness of semantic labels integration in the occupancy network encoding process.

N.B. By the fourth month of the internship, the UniTraj codebase was updated, and a new version of the paper with better baseline scores was released. We were able to reproduce this new baseline after additional discussions with the authors; as presented in Table 1. It's important to note that the consistency of our method is evident, as we continued to observe significant improvements even with the new and better baseline. This also validates the robustness and effectiveness of our approach. Unfortunately, given the remaining time, we did not reproduce these experiments again with the new baseline and focused on encoding the full point cloud directly without semantic labels and without map restriction, leading to the results presented Section 2.3.3. By the end of the internship, we aim to add predicted semantic labels using a point cloud segmentation network Puy et al. (2023) trained on the nuScenes dataset to evaluate the effect of having semantic information on the full point cloud, without any map restriction.

6 Conclusion and Future Works

In this work, we explored the potential of using occupancy in the form of LiDAR point clouds or semantic 3D volumes with object-type labels associated with each point. Using a 3D point cloud encoder capable of extracting rich semantic information in the form of 2D occupancy feature maps, we proposed two new late fusion methods directly inspired from driver behaviours for integrating occupancy into Wayformer Nayakanti et al. (2022), a state-of-the-art motion forecasting model. Both methods, particularly the explicit occupancy late fusion strategy, outperformed the baseline Wayformer across all our metrics of interest.

Furthermore, we evaluated the use of occupancy alone for motion forecasting by ablating the context agents' interaction tensor as input to Wayformer. Although this model did not outperform the baseline, it remains promising, as it achieved comparable performance to other motion forecasting models that rely on curated ground truth past trajectory inputs, notably Girgis et al. (2022).

Through further comprehensive ablation studies, we also demonstrated that having semantic labels is significant in leveraging occupancy for motion forecasting.

In **future work**, we will explore the effect of semantic labels (predicted by a semantic segmentation model [Puy et al. (2023), Puy et al. (2024)]) in the context of occupancy-only motion forecasting. We believe that this additional information in the point cloud prior to encoding will provide more fine-grained insights into the nature of object instances in the scene as well as their flow, potentially leading to state-of-the-art performance. We hope this opens the door to a new way of approaching motion forecasting in autonomous driving by directly training models from raw sensor inputs, in contrast to the current approach used by SOTA models [Nayakanti et al. (2022), Shi et al. (2023), Girgis et al. (2022)] that rely heavily on curated ground truth past trajectory inputs, which require significant annotation effort and are not available at a very large scale.

We aim to apply our approach to other state-of-the-art motion forecasting models, such as MTR Shi et al. (2023). We also plan to evaluate the impact of adding the occupancy feature map an end-to-end detection, tracking then forecasting approach Xu et al. (2024), where the past trajectory inputs are not ground truth but rather predicted bounding boxes from perception modules. These predicted bounding boxes are more likely to include false positives and false negatives in vehicle detection, errors we hope to mitigate by incorporating occupancy information.

We would also like to adopt a more explainable approach by studying what happens inside the network. This involves analyzing the attention maps of the multi-modal encoder, on the one hand, to understand which inputs truly contribute to the model's performance, and the implicitly learned offsets in the implicit occupancy late fusion approach, on the other hand, to identify which regions of the occupancy feature map are predicted as motion modes to analyze the semantic occupancy in those areas of the scene and get additional insights for the model explainability.

References

Ben Agro, Quinlan Sykora, Sergio Casas, and Raquel Urtasun. Implicit occupancy flow fields for perception and prediction in self-driving, 2023. URL https://arxiv.org/abs/2308.01471.

Anh-Quan Cao, Angela Dai, and Raoul de Charette. Pasco: Urban 3d panoptic scene completion with uncertainty awareness, 2024. URL https://arxiv.org/abs/2312.02158.

Kan Chen, Runzhou Ge, Hang Qiu, Rami AI-Rfou, Charles R. Qi, Xuanyu Zhou, Zoey Yang, Scott Ettinger, Pei Sun, Zhaoqi Leng, Mustafa Baniodeh, Ivan Bogun, Weiyue Wang, Mingxing Tan, and Dragomir Anguelov. Womd-lidar: Raw sensor dataset benchmark for motion forecasting, 2024. URL https://arxiv.org/abs/2304.03834.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks, 2019. URL https://arxiv.org/abs/1904.08755.

Lan Feng, Mohammadhossein Bahari, Kaouther Messaoud Ben Amor, Éloi Zablocki, Matthieu Cord, and Alexandre Alahi. Unitraj: A unified framework for scalable vehicle trajectory prediction, 2024. URL https://arxiv.org/abs/2403.15098.

Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction, 2022. URL https://arxiv.org/abs/2104.00563.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2019. URL https://arxiv.org/abs/1912.12180.

Quanyi Li, Zhenghao Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling, 2023. URL https://arxiv.org/abs/2306.12241.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Norman Mu, Jingwei Ji, Zhenpei Yang, Nate Harada, Haotian Tang, Kan Chen, Charles R. Qi, Runzhou Ge, Kratarth Goel, Zoey Yang, Scott Ettinger, Rami Al-Rfou, Dragomir Anguelov, and Yin Zhou. Most: Multi-modality scene tokenization for motion prediction, 2024. URL https://arxiv.org/abs/2404.19531.

Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple efficient attention networks, 2022. URL https://arxiv.org/abs/2207.05844.

Gilles Puy, Alexandre Boulch, and Renaud Marlet. Using a waffle iron for automotive point cloud semantic segmentation, 2023. URL https://arxiv.org/abs/2301.10100.

Gilles Puy, Spyros Gidaris, Alexandre Boulch, Oriane Siméoni, Corentin Sautier, Patrick Pérez, Andrei Bursuc, and Renaud Marlet. Three pillars improving vision foundation model distillation for lidar, 2024. URL https://arxiv.org/abs/2310.17504.

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement, 2023. URL https://arxiv.org/abs/2209.13508.

Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds, 2022. URL https://arxiv.org/abs/2210.07372.

Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving, 2023. URL https://arxiv.org/abs/2304.14365.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Yihong Xu, Éloi Zablocki, Alexandre Boulch, Gilles Puy, Mickael Chen, Florent Bartoccioni, Nermin Samet, Oriane Siméoni, Spyros Gidaris, Tuan-Hung Vu, Andrei Bursuc, Eduardo Valle, Renaud Marlet, and Matthieu Cord. Valeo4cast: A modular approach to end-to-end forecasting, 2024. URL https://arxiv.org/abs/2406.08113.

Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection, 2020. URL https://arxiv.org/abs/2012.11704.

A The nuScenes Dataset, a standart dataset for autonomous driving.

For this study we used nuScenes, a large-scale public dataset for autonomous driving provided by Motional and released in March 2019[8]. It was released with this aim to support the development of safe urban driving technologies. As described in their website, it includes 1000 driving scenes from Boston and Singapore, selected to depict diverse and challenging driving situations. Each scene is 20 seconds long and annotated at 2Hz with 3D bounding boxes for 23 object classes (e.g., car, truck, pedestrian, etc.). The full dataset includes approximately 1.4M camera images, 390k LiDAR sweeps, 1.4M RADAR sweeps and 1.4M object bounding boxes in 40k keyframes. Figure General visualization of the camera arrangement are show in figure 16.

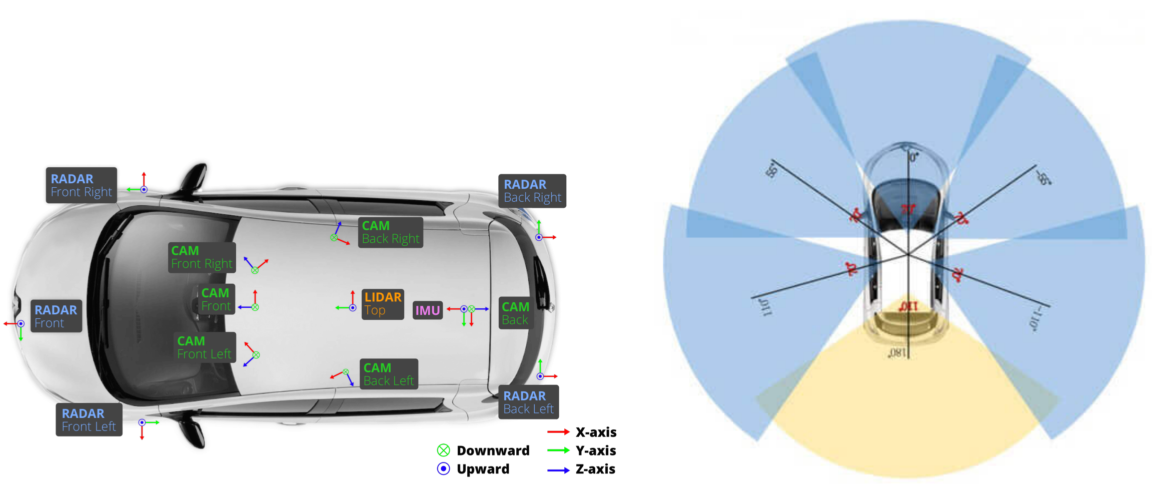


Figure 16: Camera setup for the nuScenes dataset. Left: The arrangement and orientation of the 6 camera, as well as the LiDAR sensor position are shown. Left: the field of view of each sensor is presented. For this internship we only use the LiDAR point cloud.

A.1 Dataset Processing

The nuScenes dataset contains 1,000 scenes, each spanning 20 seconds, captured at a frequency of 2 frames per second. This results in 40 frames per scene, offering a rich temporal representation of traffic scenarios. Given the goal of forecasting 6 seconds of future trajectories based on 2 seconds of past trajectory, which makes only a total of 8 seconds in the 20 seconds scene, and with each scene including multiple vehicles, the number of motion forecasting scenarios we created to train our network is $40\times$ the number of scenes in nuScenes. This augmentation is crucial for the models to achieve better generalization. Also, the frequency of the symbolic input objects data was upsampled from 2Hz to 10Hz through linear interpolation for the forecasting in the UniTraj framework.

To process and prepare the data, we used ScenarioNet Li et al. (2023), a powerful tool initially designed for traffic scenario simulation but adapted in UniTraj to unify data formats across different trajectory prediction datasets. ScenarioNet allows us to convert raw nuScenes data into a standardized

[8]https://www.nuScenes.org/nuScenes

scenario description format, facilitating seamless integration with the motion forecasting models. This format encapsulates key elements such as road graphs, agent trajectories, and metadata in a nested dictionary structure.

For each motion forecasting scenario, ScenarioNet processes the following components:

- **Map Features:** This includes lanes and lane lines, represented as polylines. Each lane has additional fields such as polygons for determining drivable areas and connectivity information to identify neighboring, preceding, and following lanes.
- **Objects:** The dataset provides an object-centric view, where each object (typically a vehicle) is described by its trajectory, velocity, and validity across time frames. This format simplifies querying and ensures that each object's motion can be efficiently tracked and predicted throughout the scene.
- **Traffic Lights:** Although nuScenes does not provide traffic light data, the ScenarioNet framework is capable of handling such information, should it be available, by associating traffic light states with relevant lanes.
- **Metadata:** Additional information such as the dataset source, time intervals between frames, and scenario statistics are included to provide context and aid in scenario selection and filtering during preprocessing.

Through this robust data preparation pipeline, we ensure that our motion forecasting models receive well-structured, consistent input data, enabling models training and evaluation.