IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

SIMPL: A Simple and Efficient Multi-agent Motion Prediction Baseline for Autonomous Driving

Lu Zhang[1], Peiliang Li[2], Sikang Liu[2], and Shaojie Shen[1]

***Abstract*—This paper presents a Simple and effIcient Motion Prediction baseLine (SIMPL) for autonomous vehicles. Unlike conventional agent-centric methods with high accuracy but repetitive computations and scene-centric methods with compromised accuracy and generalizability, SIMPL delivers real-time, accurate motion predictions for all relevant traffic participants. To achieve improvements in both accuracy and inference speed, we propose a compact and efficient global feature fusion module that performs directed message passing in a symmetric manner, enabling the network to forecast future motion for all road users in a single feed-forward pass and mitigating accuracy loss caused by viewpoint shifting. Additionally, we investigate the continuous trajectory parameterization using Bernstein basis polynomials in trajectory decoding, allowing evaluations of states and their higher-order derivatives at any desired time point, which is valuable for downstream planning tasks. As a strong baseline, SIMPL exhibits highly competitive performance on Argoverse 1 & 2 motion forecasting benchmarks compared with other state-of-the-art methods. Furthermore, its lightweight design and low inference latency make SIMPL highly extensible and promising for real-world onboard deployment. We open-source the code at https://github.com/HKUST-Aerial-Robotics/SIMPL.**

***Index Terms*—Deep Learning Methods, Intelligent Transportation Systems, Representation Learning**

I. Introduction

MOTION forecasting for the surrounding traffic participants is essential in autonomous vehicles, especially for the downstream decision-making and planning modules, since accurate and timely intention and trajectory prediction will benefit both safety and riding comfort significantly.

For learning-based motion prediction, one of the most important topics is context representation. Early approaches typically represent the surrounding scene as a multi-channel bird's-eye-view image [1]–[4]. In contrast, more recent research has increasingly embraced vectorized scene representations [5]–[13], in which the locations and geometries are annotated using point sets or polylines with geographic coordinates, leading to enhanced fidelity and expanded receptive fields. However, for both rasterized and vectorized representation, there exists a key question: how should we choose a suitable reference frame for all these elements? One straightforward way is to depict all instances within a shared coordinate system (scene-centric), such as one centered around the autonomous vehicle, and directly use the coordinates as the input features. This allows us to make predictions for multiple target agents in a single feed-forward pass [8, 14]. Yet, using global coordinates as the input, which often vary in a large span, will greatly intensify the inherent complexity of the task, resulting in degraded network performance and limited adaptability to novel scenarios. To achieve better accuracy and robustness, a common solution is normalizing the scene context w.r.t. the current state of the target agent [5, 7, 10]–[13] (agent-centric). This requires repeatedly normalizing and encoding features for each target, improving performance but increasing redundant computations. Hence, it is essential to explore an approach that can efficiently encode features for multiple targets while retaining robustness to changes of perspective.

Fig. 1: Illustration of multi-agent motion prediction in complex driving scenarios. Our method is able to generate reasonable hypotheses for all relevant agents simultaneously in a real-time fashion. The ego and other vehicles are shown in red and blue, respectively. Predicted trajectories are visualized using gradient color according to the timestamps. Please refer to the attached video for more examples.

For the downstream modules of motion prediction, such as decision-making and motion planning, it is imperative to consider not only the future positions but also the headings, velocities, and other higher-order derivatives. For example, the predicted heading of surrounding vehicles plays a pivotal role in shaping the future spatiotemporal occupancy, a critical factor in ensuring safe and robust motion planning [15, 16]. In addition, the independent anticipation of higher-order quantities without adhering to physical constraints can introduce inconsistencies in prediction outcomes [17, 18]. For instance, it may generate positional displacement despite a zero velocity, leading to confusion in planning modules.

In this paper, we propose SIMPL (**S**imple and eff**I**cient **M**otion **P**rediction base**L**ine) for autonomous driving systems, addressing critical issues in multi-agent trajectory prediction for real-world onboard applications. Firstly, we introduce the instance-centric scene representation followed by a symmetric

Manuscript received November 11, 2023; accepted February 19, 2024. This paper was recommended for publication by Editor Jens Kober upon evaluation of the reviewers' comments. This work was supported by the Hong Kong Ph.D. Fellowship Scheme, The Research Grants Council General Research Fund (RGC GRF) project RMGS20EG20, and the HKUST-DJI Joint Innovation Laboratory. *(Corresponding author: Lu Zhang.)*

[1]L. Zhang and S. Shen are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong (email: {lzhangbz, eeshaojie}@ust.hk).

[2]P. Li and S. Liu are with the DJI Technology Company, Ltd., Shenzhen, China (email: {peiliang.li, sikang.liu}@dji.com).

Digital Object Identifier (DOI): see top of this page.

fusion Transformer (SFT), enabling efficient trajectory forecasting for all agents in a single feed-forward pass, while retaining accuracy and robustness brought by the viewpoint-invariant property. Compared with other recent works based on symmetric context fusion [19]–[21], the proposed SFT is notably simpler, more lightweight, and easier to implement, making it suitable for onboard deployment.

Secondly, we introduce a novel parameterization method for the predicted trajectories based on the Bernstein basis polynomial (also known as the Bézier curve). This continuous representation ensures smoothness and enables effortless evaluation of exact states and their higher-order derivatives at any given time point. Our empirical studies indicate that learning to forecast the control points of Bézier curves is more effective and numerically stable compared to estimating the coefficients of monomial basis polynomials.

Lastly, the proposed components are well integrated into a simple and efficient model. We evaluate the proposed method on two large-scale motion forecasting datasets [22, 23], and the experimental results show that SIMPL is highly competitive when compared with other state-of-the-art methods despite its streamlined design. More importantly, SIMPL achieves efficient multi-agent trajectory prediction with fewer learnable parameters and lower inference latency without sacrificing quantitative performance, which is promising for real-world onboard deployment. We also highlight that SIMPL gains excellent extensibility as a strong baseline. The succinct architecture facilitates straightforward integration with recent advances in motion forecasting, offering opportunities for further enhancements in overall performance.

II. Related Work

A. Context Encoding and Fusion

Driving context can be broadly categorized into two main types: the historical trajectories of surrounding agents and static map information. Trajectories, as time series data, are typically encoded by temporal networks [24, 25]. As for the map features, early works commonly represent it as multi-channel bird's-eye-view images with different semantic elements rendered in distinct channels, then utilize convolutional neural networks (CNNs) to perform feature fusion [1]–[4]. However, the rasterization inevitably introduces information loss and leads to limited receptive fields. To address these issues, vectorized-based methods are proposed [5, 6] and become increasingly prevalent [7]–[13]. In such methods, map elements are represented as polylines [5, 7, 9, 11] and sparse graphs [6, 10, 13], preserving spatial information using raw coordinates. These features are further processed via graph neural networks [26, 27] or Transformers [28], yielding higher fidelity and better efficiency.

B. Symmetric Scene Modeling

Both scene-centric [8, 14] and agent-centric [5, 7, 10]–[13] representations have their limitations, necessitating a trade-off between accuracy and computational overhead. Recently, several approaches [9, 19]–[21] have emerged to address this issue by introducing symmetric modeling into the feature fusion process. HiVT [9] normalizes the local context for each agent and explicitly incorporates relative poses in both local and global feature fusion, making the method viewpoint-invariant. HDGT [19] and GoRela [20] introduce the pairwise relative positional encoding in the message passing of the heterogeneous graphs. Furthermore, QCNet [21] extends the viewpoint-invariant property to the spatial-temporal domain by incorporating the time dimension into the relative positional encoding, enabling support for streaming processing. Compared with these approaches, our work adopts a similar idea but proposes a compact symmetric fusion module, which is distinctly simpler, more lightweight, and easier to implement.

C. Trajectory Representation

Predicted trajectories are commonly represented as sequences of discrete states, such as positional coordinates [5, 6] or mixtures of probability distributions [3, 9]. Since there is no explicit constraint between discrete states, this always leads to jagged, kinematically infeasible trajectories. An alternative method predicts control signals and integrates them recurrently into trajectories according to kinematic models [29, 30]. However, this recurrent formulation tends to be less efficient and can be more susceptible to perception errors. Continuous trajectory parameterization, such as Bézier curves, is widely used in trajectory planning for mobile robots. One can efficiently generate a smooth and continuous optimal trajectory by manipulating the control points while considering certain objectives and constraints [31, 32]. In this paper, we leverage Bézier curves as the output form, which ensures single-step decoding without recurrent unrolling while maintaining better numerical stability compared to the monomial polynomials [17].

III. Methodology

A. Problem Formulation

The trajectory prediction task involves generating potential future trajectories for the target agents based on the observed motion history of moving objects and the surrounding map information. Specifically, in a driving scenario with N_a moving agents (including AV), we use $\mathcal{M}$ to represent the map information and use $\mathbf{X} = \{\mathbf{x}_0, \dots, \mathbf{x}_{N_a}\}$ to collectively denote the observed trajectories of all agents. Here, each $\mathbf{x}_i = \{x_{i,-H+1}, \dots, x_{i,0}\}$ represents the historical trajectory of the i-th agent over the past H time steps. Without loss of generality, the multi-agent motion predictor generates potential future trajectories for all N_a agents in the scene, represented as $\mathbf{Y} = \{\mathbf{y}_0, \dots, \mathbf{y}_{N_a}\}$. For each individual agent i, K possible future trajectories and their corresponding probability scores are predicted to capture the inherent multimodal distribution. The multimodal trajectories are represented as $\mathbf{y}_i = \{\mathbf{y}_i^1, \dots, \mathbf{y}_i^K\}$, where each $\mathbf{y}_i^k = \{y_{i,1}^k, \dots, y_{i,T}^k\}$ is the k-th predicted trajectory of the i-th agent over the prediction horizon T, while the probability score list is represented as $\alpha_i = \{\alpha_i^1, \dots, \alpha_i^K\}$. Hence, the multimodal prediction for agent i can be seen as estimating a mixture distribution

$$P(\mathbf{y}_i|\mathbf{X}, \mathcal{M}) = \sum_{k=1}^{K} \alpha_i^k P(\mathbf{y}_i^k|\mathbf{X}, \mathcal{M}).$$

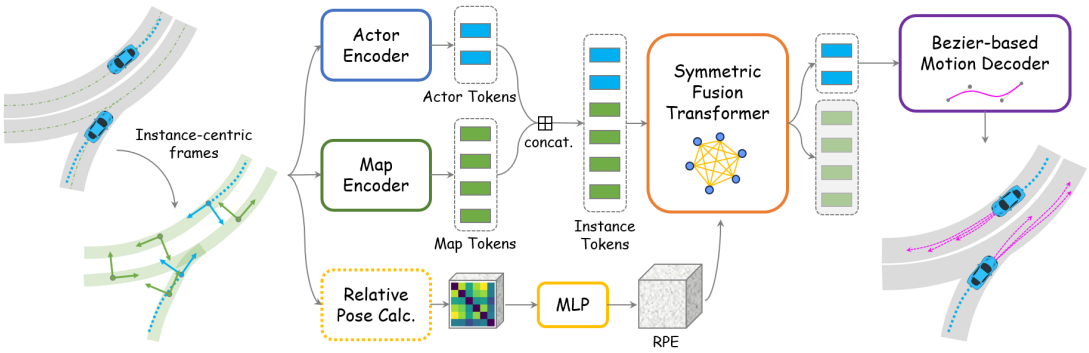

Fig. 2: Illustration of SIMPL. We utilize the simplest possible network architecture to demonstrate its effectiveness. The local features of semantic instances are processed by simple encoders, while the inter-instance features are preserved in the relative positional embeddings. Multimodal trajectory prediction results are generated by the motion decoder after the proposed symmetric feature Transformer.

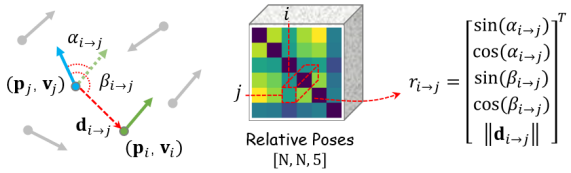

Fig. 3: Illustration of the relative pose calculation. A typical scene is depicted on the left, and we leave out the y-axis of the anchor poses for conciseness. The relative pose between instance i and j can be described by the heading difference $\alpha_{i\to j}$, relative azimuth $\beta_{i\to j}$, and positional distance $\|\mathbf{d}_{i\to j}\|$. The *all-to-all* relative poses are calculated and formulated as a 3D array.

Note that we primarily focus on marginal motion prediction in this paper, but our approach can be smoothly extended to joint motion prediction tasks by involving scene-level loss functions [8, 33]. We leave it as an important future work.

B. Framework Overview

An overview of the proposed SIMPL framework is shown in Fig. 2. Firstly, we adopt the vectorized scene representation. For each semantic instance, such as trajectories and lane segments, we construct a local reference frame to decouple inherent features and relative information between instances. Then, the actor and map features are extracted by simple encoders, while the relative poses of instances are calculated in pairs and further encoded by a multilayer perceptron (MLP) to get the relative positional embedding (RPE). Instance tokens and RPE are then sent into the proposed symmetric fusion Transformer (SFT), a compact and succinct fusion module that symmetrically updates features. Finally, the Bézier curve parameterized trajectories are predicted by a simple decoder for all target agents simultaneously.

C. Instance-centric Scene Representation

Apart from scene-centric representation, a scenario can be represented by vectorized features under the local frames of the instances, along with the relative poses between them. A local reference frame is established for each semantic element to normalize its spatial attributes, which we term as "instance-centric". Without loss of generality, we locate the reference frame at the current observed state for agents' historical trajectories. Regarding map elements, such as lane segments, we use the centroid of the polyline as the anchor point and employ the displacement vector between endpoints as the heading angle. Intuitively, local coordinate frames can be seen as "anchor poses" of instances, therefore, the relative spatial information can be easily calculated pair-by-pair.

Specifically, the anchor pose under a global coordinate frame for element i can be represented using its position $\mathbf{p}_i \in \mathbb{R}^2$ and heading vector $\mathbf{v}_i \in \mathbb{R}^2$. Following [20], we describe the relative pose between element i and element j using three quantities: heading difference $\alpha_{i\to j}$, relative azimuth $\beta_{i\to j}$, and distance $\|\mathbf{d}_{i\to j}\|$. To enhance numerical stability, angles are represented using their sine and cosine values. We denote the heading difference $\alpha_{i\to j}$ as

$$\sin(\alpha_{i\to j}) = \frac{\mathbf{v}_i \times \mathbf{v}_j}{\|\mathbf{v}_i\|\|\mathbf{v}_j\|}, \quad \cos(\alpha_{i\to j}) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\|\|\mathbf{v}_j\|},$$

and the relative azimuth $\beta_{i\to j}$ (the angle between displacement vector $\mathbf{d}_{i\to j} = \mathbf{p}_i - \mathbf{p}_j$ and heading vector $\mathbf{v}_j$) as

$$\sin(\beta_{i\to j}) = \frac{\mathbf{d}_{i\to j} \times \mathbf{v}_j}{\|\mathbf{d}_{i\to j}\|\|\mathbf{v}_j\|}, \quad \cos(\beta_{i\to j}) = \frac{\mathbf{d}_{i\to j} \cdot \mathbf{v}_j}{\|\mathbf{d}_{i\to j}\|\|\mathbf{v}_j\|}.$$

For simplicity, we omit the additional positional encoding process for the distance value used in [20], making the relative spatial information a 5-dimensional vector $r_{i\to j} = [\sin(\alpha_{i\to j}), \cos(\alpha_{i\to j}), \sin(\beta_{i\to j}), \cos(\beta_{i\to j}), \|\mathbf{d}_{i\to j}\|]$. We can conveniently calculate the *all-to-all* relative spatial information by leveraging the broadcasting mechanism of PyTorch or NumPy. Consequently, given a scene contains $N = N_a + N_m$ semantic elements, the resulting relative positional info is an array with the shape of $[N, N, 5]$, while $r_{i\to j}$ locates at the j-th row and i-th column. An illustration of the relative pose calculation is shown in Fig 3.

D. Context Feature Encoding

After obtaining the instance-centric representation and relative positional encoding for instances, we utilize corresponding encoders (also serving as "tokenizers") to convert them into feature vectors. To keep SIMPL simple, we use the 1D CNN-based network [6] for handling historical trajectories and employ a PointNet-based encoder [5, 34] for extracting static map features. Without loss of generality, we let all latent features have D channels. Therefore, the resulting actor and map tokens have the shapes of $[N_a, D]$ and $[N_m, D]$, where N_a is the number of actors and N_m is the number of map elements. For the detailed implementation, we refer readers to [5, 6]. Moreover, the relative pose encoding is further encoded by an MLP, yielding the relative positional embedding (RPE) with the shape of $[N, N, D]$.

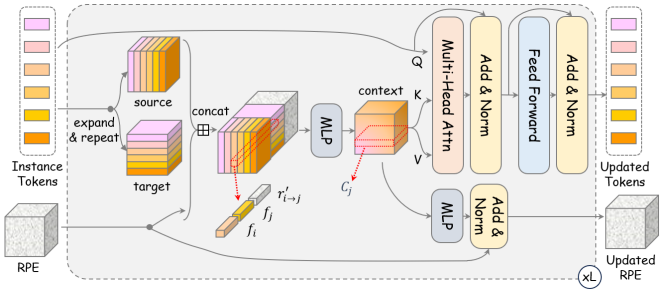

Fig. 4: Illustration of the proposed symmetric fusion Transformer (SFT) with L layers. Instance tokens and RPE are recurrently updated in each SFT layer.

E. Symmetric Fusion Transformer

Once the instance tokens and the corresponding RPE are obtained, we employ the proposed symmetric fusion Transformer (SFT) to update the instance tokens in a viewpoint-invariant manner. Fig. 4 shows the overall structure of the proposed SFT, which comprises multiple stacked SFT layers, akin to the standard Transformer [28]. In essence, we can view the driving scene as a complete digraph with self-loops, in which the input instance-centric features serve as nodes and the RPE depicts the edge information. During the update process, the node features are influenced solely by the graph edges associated with the target node, ensuring that the feature fusion remains viewpoint-invariant.

In the *microscopic* view, we denote the token of i-th and j-th instances as f_i and f_j, respectively. And the RPE vector associated with the edge from f_i to f_j is designated as $r'_{i\to j}$. The tuple $\left(f_i, f_j, r'_{i\to j}\right)$ encompasses all information intended to be transmitted from node i to node j, therefore, we can employ a simple MLP to encode these features and get the i-th context vector for node j

$$c_{i\to j} = \phi\left(f_i \boxplus f_j \boxplus r'_{i\to j}\right),$$

where $\boxplus$ denotes the concatenation operator, and $\phi : \mathbb{R}^{3D} \to \mathbb{R}^D$ denotes the MLP, which consists of a linear layer, layer normalization, and ReLU activation. We then perform cross-attention on the target node and its context,

$$f'_j = \texttt{MHA}(\texttt{Query} : f_j, \texttt{Key} : C_j, \texttt{Value} : C_j),$$

where $\texttt{MHA}(\cdot,\cdot,\cdot)$ is the standard multi-head attention function, and $C_j = \{c_{i\to j}\}_{i\in\{1,\dots,N\}}$ is the set of context vectors of token j. Note that C_j contains $c_{j\to j}$ as well, indicating the presence of a self-loop for each node. Similar to the standard Transformer, a feed-forward layer is joined after the attention module. Besides, in each layer, $r'_{i\to j}$ is updated by re-encoding the context vector using another MLP and subsequently added to the input RPE through the residual connection.

In practice, we provide a more efficient implementation of the aforementioned feature fusion in a vectorized way (see Fig. 4). Firstly, given the input instance tokens $F \in \mathbb{R}^{N\times D}$, we expand it along different dimensions and replicate it N times to build the source and target node arrays, which both exhibit the shape of $[N, N, D]$. After concatenation of the source array, target array, and the corresponding RPE, the array of tuple $\left(f_i, f_j, r'_{i\to j}\right)$ is obtained, and we apply ϕ to get the context array $C \in \mathbb{R}^{N\times N\times D}$. Note that the j-th row of C is exactly C_j, representing the collection of context features

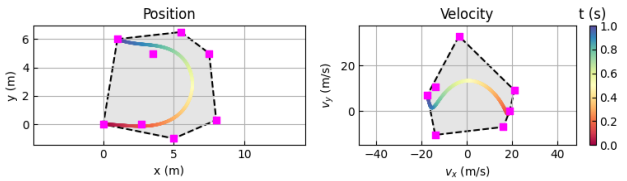

Fig. 5: A 2D septic Bézier curve (left). Pink dots are control points while grey polygons are corresponding convex hulls. When the time duration of the trajectory is 1 second, the 1st-order derivative will exactly be the velocity profile (right), which is also a Bézier curve due to the hodograph property.

centered around token j. Hence, we employ C for the key and value, while the expanded $F \in \mathbb{R}^{N\times 1\times D}$ serves as the query. The standard multi-head attention module then passes messages from context features to instance tokens. The rest of the SFT layer also enjoys the vectorized implementation, but we won't delve into the details due to its simplicity. Note that our proposed SFT layer shares similarities with recent "query-centric" methods [21, 35], but we incorporate global attention and RPE updates, resulting in a more compact design. For the detailed implementation, please refer to the released code.

F. Multimodal Continuous Trajectory Decoder

After the symmetric global feature fusion, the updated actor tokens are gathered and sent to a multimodal motion decoder to generate predictions for all agents. Here, we forecast K possible futures, and for each mode, a simple MLP is applied, which has a regression head for trajectories and a classification head followed by a softmax function for their corresponding probability scores.

As for the trajectory regression head, in contrast to previous approaches that directly predict the positions of future trajectories, we choose to use the continuous parameterized representation. Parameterized curves (e.g., polynomials) bring a continuous representation, which allows for obtaining smooth motion and exact high-order derivatives at any time point. However, according to the previous studies [30], the monomial basis polynomial representation hurts performance significantly. We blame the degeneration on the numerical imbalance of the predicted coefficients (see Sec. IV-C2 for details), making the regression a hard task.

To leverage the advantage of parametric trajectory while avoiding performance degeneration, we introduce the Bernstein basis polynomial (i.e., Bézier curve), of which coefficients are control points with concrete spatial meaning, resulting in better convergence. Specifically, an degree n Bézier curve is written as

$$f(t) = \sum_{i=0}^{n} b_n^i(t) p_i = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} p_i,\ t \in [0,1],$$

where $b_n^i(t)$ is the i-th order Bernstein basis, $\binom{n}{i}$ is the binomial coefficient, t is the variable of the parametric curve, and p_i is the control point. Note that for an n-th order Bézier curve, there are $n+1$ control points in total, and the first and last control points are always the endpoints of the curve (see Fig. 5). As Bézier curves are defined on $t \in [0,1]$, we normalize the actual time $\tau \in [0, \tau_{max}]$, so the parametric curve can be written as $f(t) = \sum_{i=0}^{n} b_n^i(\frac{\tau}{\tau_{max}}) p_i$. Moreover, due to the hodograph property, the derivative of an nth-order

Bézier curve is still a Bézier curve, with control points defined by $p_i^{(1)} = n(p_{i+1} - p_i)$, namely, the velocity profile of the trajectory can be calculated by

$$f'(t) = \frac{n}{\tau_{max}} \sum_{i=0}^{n-1} b_{n-1}^i(\frac{\tau}{\tau_{max}})(p_{i+1} - p_i),\ \tau \in [0, \tau_{max}].$$

In practice, we use a simple MLP as the regression head that performs the mapping from the fused actor features to the control points. Then, the positional coordinates of each predicted trajectories $\mathbf{Y}_{pos} \in \mathbb{R}^{T\times 2}$ can be simply calculated by multiplying the constant basis matrix $\mathbf{B} \in \mathbb{R}^{T\times(n+1)}$ and the corresponding predicted 2D control points $\mathbf{P} \in \mathbb{R}^{(n+1)\times 2}$ (independent for x-axis and y-axis),

$$\begin{aligned}\mathbf{Y}_{pos} &= \mathbf{B} \times \mathbf{P} \\ &= \begin{bmatrix} b_n^0(t_1) & b_n^1(t_1) & ... & b_n^n(t_1) \\ \vdots & \vdots & & \vdots \\ b_n^0(t_T) & b_n^1(t_T) & ... & b_n^n(t_T) \end{bmatrix} \begin{bmatrix} p_0^x & p_0^y \\ \vdots & \vdots \\ p_n^x & p_n^y \end{bmatrix},\end{aligned}$$

where T is the number of sampled timestamps required for the predicted trajectories, and $t_i = \frac{\tau_i}{\tau_{max}}$ is the normalized time point. We also point out that the velocity and other higher-order derivatives of the predicted trajectories can be retrieved by the similar procedures mentioned above and we omit it for conciseness. For agents with non-holonomic constraints, like vehicles and cyclists, the yaw angle aligns with the tangent vector of the trajectory, and we can derive the heading angle for each state from the velocity estimation. Finally, predicted trajectories are further transformed back to the global frame according to the corresponding anchor pose of the actors.

G. Training

The proposed SIMPL is trained in an end-to-end manner. The overall loss function is the weighted sum of the regression loss and classification loss

$$\mathcal{L} = \omega\mathcal{L}_{reg} + (1 - \omega)\mathcal{L}_{cls},$$

where $\omega \in [0, 1]$ is the weight to balance these components, and we set $\omega = 0.8$ to address the importance of the regression task. Following [6], we use the winner-takes-all (WTA) strategy for handling the multimodality. For each agent, we find the best-predicted trajectory k^* among the K hypotheses by picking the one with minimum final displacement error. Regarding the classification task, we use the max-margin loss to distinguish the positive mode from others similar to [6]. For the trajectory regression task, in addition to positional coordinate regression, we introduce an optional yaw angle loss to provide auxiliary supervision, resulting in

$$\mathcal{L}_{reg} = \texttt{PosLoss}(\bar{Y}_{pos}, Y_{pos}^{k^*}) + \texttt{YawLoss}(\bar{Y}_{yaw}, Y_{yaw}^{k^*}),$$

where $\bar{Y}_{(\cdot)}$ denotes the ground truth (GT) states, and $Y_{(\cdot)}^{k^*}$ is the predicted position and yaw angle of the winner mode. We employ the smooth L1 loss as the position regression loss, and we designate the yaw regression loss as

$$\texttt{YawLoss}(\bar{Y}_{yaw}, Y_{yaw}^{k^*}) = [1 - \texttt{CosSim}(\bar{Y}_{yaw}, Y_{yaw}^{k^*})]/2,$$

where $\texttt{CosSim}(\cdot, \cdot)$ is the cosine similarity measurement, which yields a value of 1 for two aligned yaw vectors and a value of -1 for two opposite yaw vectors. Incorporating yaw angle loss implicitly strengthens the consistency between consecutive states, making predicted trajectories with higher smoothness and kinematic feasibility, and resulting in more realistic trajectories, especially for low-speed agents.

IV. Experimental Results

A. Experiment Setup

1) Dataset: We evaluate the proposed method on both Argoverse 1 [22] and Argoverse 2 [23] motion forecasting datasets. Argoverse 1 contains 205942, 39472, and 78143 sequences for training, validation, and testing, respectively. Each sequence is sampled at 10 Hz, and the task involves predicting future 3-second trajectories based on 2 seconds of historical observations (i.e., $H = 20$, $T = 30$). In the case of Argoverse 2, it comprises 200000, 25000, and 25000 sequences for training, validation, and testing. The sequences are also sampled at 10 Hz, and the given history is 5 seconds while the future motion is 6 seconds (i.e., $H = 50$, $T = 60$). Both Argoverse 1 and Argoverse 2 provide HD maps.

2) Metrics: We mainly follow the standard metrics that are commonly used in multimodal trajectory prediction, including minimum average displacement error (minADE$_k$), minimum final displacement error (minFDE$_k$), miss rate (MR$_k$), and brier-minFDE$_k$. All these metrics evaluate the best-predicted trajectory for a single target agent among the K hypotheses against the ground truth. The minADE$_k$ is the average Euclidean distance between the predicted trajectory and GT, while minFDE$_k$ only considers the error at endpoints. The MR is the percentage of sequences where the obtained minFDE$_k$ is greater than 2 meters. Brier-minFDE$_k$ adds an additional brier score $(1 - p)^2$ to minFDE$_k$, where p denotes the probability of the best-predicted trajectory. We refer interested readers to [22, 23] for detailed definitions.

3) Implementation details: We set the dimension $D = 128$ for all latent vectors, and stack 4 SFT layers and 8 attention heads for the symmetric global feature fusion. For the multi-modal decoder, we set the number of modes $K = 6$ following the common setups. The degree of Bézier curve n is configured as 5 for Argoverse 1 and 7 for Argoverse 2 due to the different prediction horizons. SIMPL is trained in an end-to-end manner using a batch size of 128 for 50 epochs on a server with 8 Nvidia RTX 3090 GPUs. We employ the Adam optimizer and set the learning rate to 1e-3 in the beginning and gradually decrease it to 1e-4 after 40 epochs.

B. Results

1) Comparison with the state-of-the-art: We compare SIMPL with other state-of-the-art methods on two large-scale motion forecasting benchmarks. Tab. I shows the quantitative results of the Argoverse 1 test split. The upper part presents the single-model results, while the lower part shows the performance of methods with ensemble techniques. With such a simple design, SIMPL achieves highly competitive

Table I: Results on the test split of Argoverse 1 motion forecasting dataset. The upper and lower groups are the results of single model and ensemble methods. The best result is in **bold** while the second best result is underlined. b-minFDE$_6$ is the official ranking metric. ♯ denotes the model size is from the non-official implementation

Method	minADE$_6$	minFDE$_6$	MR$_6$	b-minFDE$_6$	#Param
LaneGCN [6]	0.870	1.362	16.2	2.053	3.7M
mmTrans [7]	0.844	1.338	15.4	2.033	2.6M
SceneTrans [8]	0.803	1.232	12.6	1.887	15.3M
HiVT [9]	**0.774**	**1.169**	12.7	1.842	2.5M
MacFormer [12]	0.819	1.216	**12.1**	1.827	2.4M
SIMPL (w/o ens)	0.793	1.179	12.3	**1.809**	**1.8M**
MultiPath++ [30]	0.790	1.214	13.2	1.793	21.1M♯
MacFormer [12]	0.812	1.214	12.7	1.767	2.4M
HeteroGCN [13]	0.789	1.160	11.7	1.751	-
Wayformer [36]	**0.768**	1.162	11.9	**1.741**	11.2M♯
SIMPL (w/ ens)	0.769	**1.154**	**11.6**	1.746	**1.8M**

Table II: Results on the Argoverse 2 test split for methods based on symmetric scene modeling. The results are from single models (w/o ensemble). The best and the second-best results are in **bold** and underlined, respectively. b-minFDE$_6$ is the official ranking metric.

Method	minADE$_6$	minFDE$_6$	MR$_6$	b-minFDE$_6$	#Param
HDGT [19]	0.84	1.60	21.0	2.24	12.1M
GoRela [20]	0.76	1.48	22.0	2.01	-
QCNet [21]	**0.65**	**1.29**	**16.0**	**1.91**	7.3M
SIMPL (w/o ens)	0.72	1.43	19.2	2.05	**1.9M**

results among all listed methods. LaneGCN [6], mmTransformer [7], and MacFormer [12] utilize the agent-centric representation, which hinders efficient online inference. Scene Transformer [8] adopts a scene-centric representation, enabling single-pass multi-agent motion prediction. However, it exhibits a larger model size and inferior performance, indicating it is data-hungry and less generalizable. HiVT [9] explicitly considers relative poses during feature fusion for robustness against viewpoint shifting, whereas SIMPL yields a simpler and lighter design while achieving better performance. We also report the evaluation results with ensembling for a fair comparison. After ensembling of 8 models based on k-means clustering, SIMPL outperforms strong baselines like MultiPath++ [30] and is competitive to state-of-the-art methods such as Wayformer [36] with much fewer parameters. The evaluation results of the Argoverse 2 motion forecasting benchmark are shown in Tab. II. We compare SIMPL with other state-of-the-art methods that employ symmetric scene modeling techniques. Characterized by its minimalist architecture and remarkably compact model size, SIMPL attains competitive trajectory prediction results and is promising for further extensions and applications.

2) Inference latency: The evaluation results of inference latency are shown in Fig. 6. All experiments are conducted on an RTX 3060Ti GPU with the original PyTorch implementation. Firstly, we compare the computational efficiency with LaneGCN [6] and HiVT [9]. As the agent-centric baseline, LaneGCN normalizes the scene w.r.t. each target's state and organizes contexts into a batched form. Both HiVT and our SIMPL employ shared context encoding and perform multi-agent prediction in one forward pass, meaning that the batch size can be set as 1. Benefiting from the compact design, SIMPL achieves real-time performance and slightly

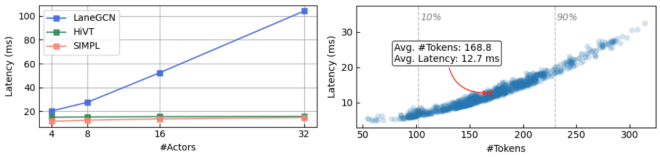

Fig. 6: Evaluation results of the inference latency on Argoverse 1 dataset. Left: Average inference latency of different methods w.r.t. the number of target agents. Both HiVT and SIMPL attain real-time performance for multi-agent motion prediction, while agent-centric approaches are hard to scale up. Right: Relation between runtime and number of total instance tokens in the scenes. Each point represents a driving scene and all of them can be processed in real-time.

Table III: Ablative study of the feature fusion module design on the Argoverse 1 validation split.

Model	Emb. Size	# Layers	RPE Upd.	minFDE$_6$	MR$_6$	b-minFDE$_6$
$\mathcal{M}1$	64	2	✗	1.237	12.8	1.848
$\mathcal{M}2$	64	4	✗	1.037	9.5	1.658
$\mathcal{M}3$	128	4	✗	0.993	9.0	1.607
$\mathcal{M}4$	128	4	✓	0.947	**8.1**	1.559
$\mathcal{M}5$	128	6	✓	**0.944**	8.4	**1.558**

better inference speed than HiVT. Besides, methods based on symmetric scene modeling enable much more efficient multi-agent prediction than conventional agent-centric counterparts. The right part of Fig. 6 shows the latency distribution on 1,000 scenes randomly sampled from the full validation set. Without any acceleration techniques like quantization, SIMPL attains high inference speed and is promising for real-world onboard deployment after further optimization.

3) Qualitative Results: The qualitative results on both Argoverse 1 and 2 datasets are illustrated in Fig. 7. Our SIMPL is able to anticipate realistic, reasonable, and accurate multimodal future trajectories for multiple agents in the scene simultaneously. We also demonstrate the qualitative results of real-time consecutive trajectory prediction on the Argoverse tracking dataset based on models trained on the motion forecasting dataset without fine-tuning (zero-shot transfer). The snapshots are depicted in Fig. 1, and for detailed results please refer to the attached supplementary video.

C. Ablation Study

1) On feature fusion module: We first investigate the design of the proposed SFT layer. As shown in Tab. III, with the growth of embedding size and number of SFT layers ($\mathcal{M}1 \rightarrow \mathcal{M}3$), SIMPL achieves better performance in all metrics. However, given the embedding size of 128, increasing the number of SFT layers from 4 to 6 ($\mathcal{M}4 \rightarrow \mathcal{M}5$) marginally improves the prediction accuracy at the cost of incorporating 22% more parameters, which is less preferred in real-time applications. We also find that updating the relative positional embedding (RPE) using the context array in each layer can boost the overall performance significantly ($\mathcal{M}3 \rightarrow \mathcal{M}4$). We surmise this is due to the fact that updating the RPE involves incorporating node features into edge features, which helps to learn the relationship between different semantic instances.

2) On trajectory parameterization: We further show the influence of different trajectory parameterization methods in Tab. IV. Similar to the conclusion described in [30], the monomial basis polynomial representation brings a significant performance drop compared with raw coordinates. In contrast,

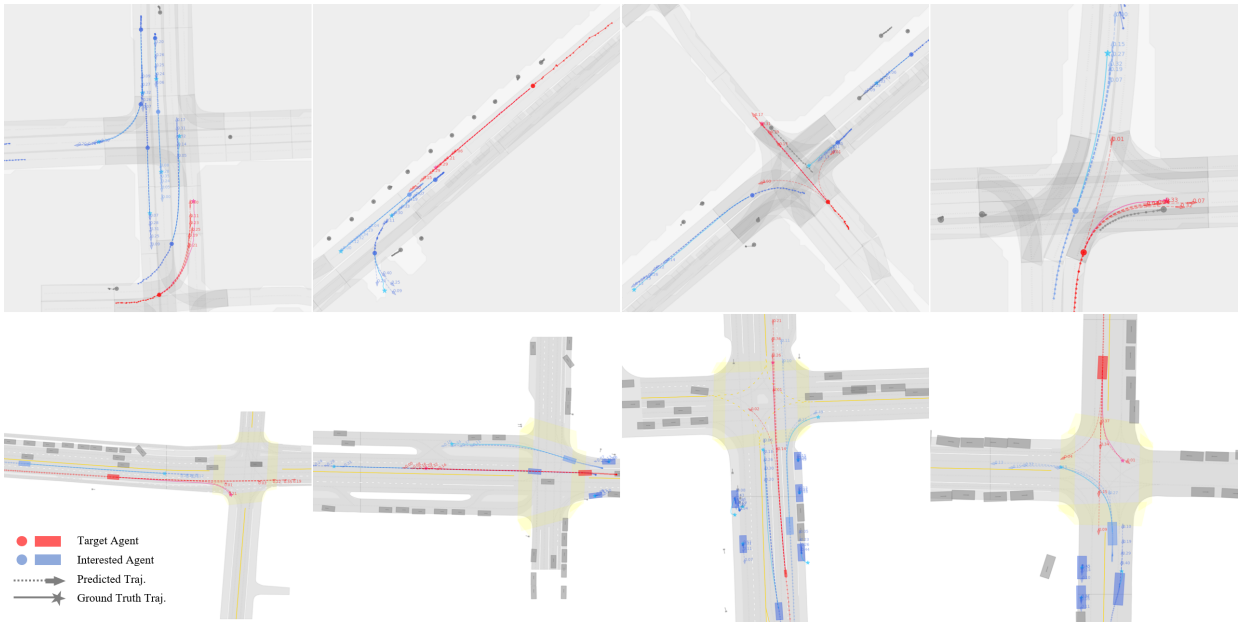

Fig. 7: Qualitative results on Argoverse 1 (upper) and Argoverse 2 (lower) motion forecasting datasets. The target agents are shown in red while other interested agents are marked in blue. Note that the future motions of all agents in the scene are generated but we omit the results of ignored agents (grey) for conciseness. The ground truth endpoints are denoted as stars and the predicted trajectories are depicted as dashed curves with the final poses marked as arrows. SIMPL effectively extracts driving context features and generates multiple agent trajectories that adhere to the specific scene constraints in complex scenarios. (Please zoom in for details.)

Table IV: Ablative study of the trajectory parameterization methods and the yaw angle loss on the Argoverse 2 validation set.

Parameterization	Yaw loss	minADE_6	minFDE_6	minAYE_6	minFYE_6
Raw coords	✗	**0.780**	**1.452**	0.134	0.151
Polynomial	✗	0.861	1.738	0.146	0.278
Bézier curve	✗	**0.780**	1.457	0.137	0.297
Bézier curve	✓	0.783	**1.452**	**0.055**	**0.076**

our Bézier curve-based method achieves the same level of results in the displacement-related metrics. We blame the performance drop on the numerical imbalance of the monomial basis, while the coefficients of Bézier curves are control points with specific spatial meanings, without significant difference in difficulty from directly regressing coordinates. We compare the predicted coefficient distributions for different parameterization methods (see Fig. 8), and it shows the distribution of Bézier curve is more regular than the monomial basis, potentially making this task easier.

3) On auxiliary loss functions: Leveraging continuous representations also makes it more convenient for us to access higher-order physical quantities without violating physical constraints. Therefore, we can naturally introduce loss functions for quantities such as heading angles without any modification to the network architecture. To evaluate the yaw loss introduced in III-G, we introduce the minimum average yaw error (minAYE_k) and minimum final yaw error (minFYE_k), which directly calculate the absolute angular difference in radians. From Tab. IV, we can clearly find that yaw loss substantially improves the accuracy of yaw angles, which is highly favorable to real-world applications.

D. Extensibility

As described above, our SIMPL follows the simplest possible network architecture design, leaving spaces for further extensions. To demonstrate this merit, we attach an additional

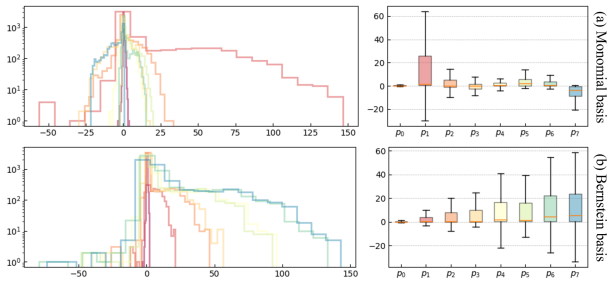

Fig. 8: Distribution of the predicted x-axis coefficients of monomial basis polynomials (a) and Bernstein basis polynomials (b). The left column shows the histogram of the coefficient distribution while the right column shows the corresponding boxplot. The distributions of coefficients of different orders are indicated by different colors. We can find that the 1st-order coefficient of the monomial basis polynomial has a much wider span than others, while the coefficients (i.e., control points) of the Bézier curves are more regular due to their specific spatial meaning.

simple trajectory decoder following the idea of iterative proposal refinement, which is widely adopted in recent state-of-the-art approaches [11, 21, 37]. We apply no modification to the vanilla SIMPL and regard the predicted multimodal trajectories as initial proposals. For simplicity, here we only refine the predicted trajectories of the target agent. Similar to [11, 37], for each proposal trajectory, we first re-encode it into a feature vector. Then, we collect the nearby instance features of each proposal within a certain range, followed by a feature fusion module based on the standard Transformer decoder, attending the local context to the corresponding proposal feature. After the refinement, the proposal features are sent to another simple MLP-based decoder to get the final predicted trajectories and their probability scores. The training process is identical to the vanilla SIMPL, namely, employing the WTA strategy with the best-predicted proposal as the positive trajectory. We denote the enhanced model as SIMPL-R, and the results are shown in Tab. V. With such a simple plug-and-play post-refinement module, SIMPL-R achieves better overall performance, indicating the compact

Table V: Quantitative results of extensibility experiment on the Argoverse 1 validation and test split.

Split	Method	minADE_6	minFDE_6	MR_6	b-minFDE_6
Val	SIMPL	0.658	0.947	**8.1**	1.559
	SIMPL-R	**0.651**	**0.946**	8.2	**1.542**
Test	SIMPL	0.793	1.179	12.3	1.809
	SIMPL-R	**0.783**	**1.173**	**12.1**	**1.781**

architecture makes it scalable and promising to be used as a backbone for a variety of different tasks. We also note that SIMPL can be smoothly integrated with other recent techniques such as self-supervised learning [38, 39]. We leave it as another future work.

V. Conclusion

In this paper, we present a simple and efficient multi-agent motion prediction baseline for autonomous driving. Leveraging the proposed symmetric fusion Transformer, the proposed method achieves efficient global feature fusion and retains robustness against viewpoint shifting. The continuous trajectory parameterization based on Bernstein basis polynomials provides higher compatibility with downstream modules. The experimental results on large-scale public datasets show that SIMPL is more advantageous in terms of model size and inference speed while obtaining the same level of accuracy as other state-of-the-art methods.

References

[1] H. Cui, V. Radosavljevic *et al.*, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2019, pp. 2090–2096.

[2] T. Zhao, Y. Xu *et al.*, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12 126–12 134.

[3] Y. Chai, B. Sapp *et al.*, "MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv preprint arXiv:1910.05449*, 2019.

[4] T. Phan-Minh, E. C. Grigore *et al.*, "CoverNet: Multimodal behavior prediction using trajectory sets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14 074–14 083.

[5] J. Gao, C. Sun *et al.*, "VectorNet: Encoding HD maps and agent dynamics from vectorized representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11 525–11 533.

[6] M. Liang, B. Yang *et al.*, "Learning lane graph representations for motion forecasting," in *Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 541–556.

[7] Y. Liu, J. Zhang *et al.*, "Multimodal motion prediction with stacked transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7577–7586.

[8] J. Ngiam, B. Caine *et al.*, "Scene Transformer: A unified architecture for predicting multiple agent trajectories," *arXiv preprint arXiv:2106.08417*, 2021.

[9] Z. Zhou, L. Ye *et al.*, "HiVT: Hierarchical vector transformer for multi-agent motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8823–8833.

[10] L. Zhang, P. Li *et al.*, "Trajectory prediction with graph-based dual-scale context fusion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, IEEE, 2022, pp. 11 374–11 381.

[11] S. Shi, L. Jiang *et al.*, "Motion transformer with global intention localization and local movement refinement," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 6531–6543, 2022.

[12] C. Feng, H. Zhou *et al.*, "Macformer: Map-agent coupled transformer for real-time and robust trajectory prediction," *IEEE Robot. Autom. Lett.*, 2023.

[13] X. Gao, X. Jia *et al.*, "Dynamic scenario representation learning for motion forecasting with heterogeneous graph convolutional recurrent networks," *IEEE Robot. Autom. Lett.*, vol. 8, no. 5, pp. 2946–2953, 2023.

[14] S. Casas, C. Gulino *et al.*, "Implicit latent variable model for scene-consistent motion forecasting," in *Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 624–641.

[15] M. Werling, S. Kammel *et al.*, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Intl. J. Robot. Res.*, vol. 31, no. 3, pp. 346–359, 2012.

[16] W. Ding, L. Zhang *et al.*, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2997–3004, 2019.

[17] T. Buhet, E. Wirbel *et al.*, "PLOP: Probabilistic polynomial objects trajectory planning for autonomous driving," *arXiv preprint arXiv:2003.08744*, 2020.

[18] X. Jia, L. Chen *et al.*, "Towards capturing the temporal dynamics for trajectory prediction: a coarse-to-fine approach," in *Conf. Robot Learn.*, PMLR, 2023, pp. 910–920.

[19] X. Jia, P. Wu *et al.*, "HDGT: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[20] A. Cui, S. Casas *et al.*, "GoRela: Go relative for viewpoint-invariant motion forecasting," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2023, pp. 7801–7807.

[21] Z. Zhou, J. Wang *et al.*, "Query-centric trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17 863–17 873.

[22] M.-F. Chang, J. Lambert *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8748–8757.

[23] B. Wilson, W. Qi *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," *arXiv preprint arXiv:2301.00493*, 2023.

[24] A. Alahi, K. Goel *et al.*, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 961–971.

[25] A. Vemula, K. Muelling *et al.*, "Social attention: Modeling attention in human crowds," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2018, pp. 4601–4607.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[27] P. Veličković, G. Cucurull *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[28] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[29] H. Cui, T. Nguyen *et al.*, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2020, pp. 10 563–10 569.

[30] B. Varadarajan, A. Hefny *et al.*, "MultiPath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2022, pp. 7814–7821.

[31] F. Gao, W. Wu *et al.*, "Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2018, pp. 344–351.

[32] S. Deolasee, Q. Lin *et al.*, "Spatio-temporal motion planning for autonomous vehicles with trapezoidal prism corridors and bézier curves," in *Proc. Amer. Control Conf.*, IEEE, 2023, pp. 3207–3214.

[33] R. Girgis, F. Golemo *et al.*, "Latent variable sequential set transformers for joint multi-agent motion prediction," *arXiv preprint arXiv:2104.00563*, 2021.

[34] C. R. Qi, H. Su *et al.*, "PointNet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[35] S. Shi, L. Jiang *et al.*, "MTR++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *arXiv preprint arXiv:2306.17770*, 2023.

[36] N. Nayakanti, R. Al-Rfou *et al.*, "Wayformer: Motion forecasting via simple & efficient attention networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, IEEE, 2023, pp. 2980–2987.

[37] S. Choi, J. Kim *et al.*, "R-Pred: Two-stage motion prediction via tube-query attention-based trajectory refinement," *arXiv preprint arXiv:2211.08609*, 2022.

[38] P. Bhattacharyya, C. Huang *et al.*, "SSL-Lanes: Self-supervised learning for motion forecasting in autonomous driving," in *Conf. Robot Learn.*, PMLR, 2023, pp. 1793–1805.

[39] J. Cheng, X. Mei *et al.*, "Forecast-MAE: Self-supervised pre-training for motion forecasting with masked autoencoders," *arXiv preprint arXiv:2308.09882*, 2023.