

Joint 3D Tracking and Forecasting with Graph Neural Network and Diversity Sampling

Xinshuo Weng*, Ye Yuan*, and Kris Kitani

Robotics Institute, Carnegie Mellon University, USA
 {xinshuw, yyuan2, kkitani}@cs.cmu.edu

Abstract. 3D multi-object tracking (MOT) and trajectory forecasting are two critical components in modern 3D perception systems that require accurate modeling of multi-agent interaction. We hypothesize that it is beneficial to unify both tasks under one framework in order to learn a shared feature representation of agent interaction. To evaluate this hypothesis, we propose a unified solution for 3D MOT and trajectory forecasting which also incorporates two additional novel computational units. First, we propose a feature interaction technique by introducing Graph Neural Networks (GNNs) to capture the way in which multiple agents interact with one another. The GNN is able to model complex hierarchical interactions, improve the discriminative feature learning for MOT association, and provide socially-aware context for trajectory forecasting. Second, we use a diversity sampling function to improve the quality and diversity of our forecasted trajectories. The learned sampling function is trained to efficiently extract a variety of outcomes from a generative trajectory distribution and helps avoid the problem of generating many duplicate trajectory samples. We evaluate on the KITTI and nuScenes datasets, showing that our unified method with feature interaction and diversity sampling achieves new state-of-the-art performance on both 3D MOT and trajectory forecasting. Our code will be made available at <https://github.com/xinshuweng/GNNTrkForecast>.

Keywords: 3D multi-object tracking, multi-agent trajectory forecasting, graph neural networks, diversity sampling.

1 Introduction

3D multi-object tracking (MOT) and trajectory forecasting are two critical components of modern 3D perception systems [60,68] in autonomous driving [47,65,8] and assistive robots [34,49]. Historically, 3D MOT [67,74,57,52,19] and trajectory forecasting [1,27,38,56,33,41,10,17,44] have been studied separately. As a result, modern perception systems often perform 3D MOT and trajectory forecasting separately in a cascaded order, where the tracking is performed first to obtain trajectories in the past, followed by the trajectory forecasting module to predict trajectories in the future. However, this system pipeline without any error feedback mechanism may not be optimal as the performance of tracking and forecasting modules highly depends on each other.

* Contributed equally to this work.

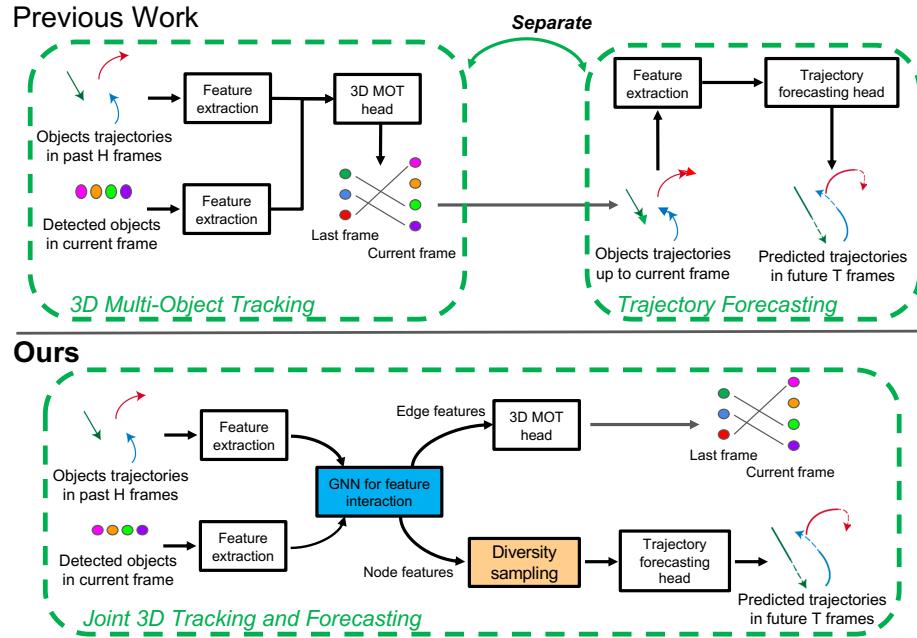


Fig. 1: **(Top)** Previous work has studied 3D MOT and trajectory forecasting separately. The entire pipeline is in a cascaded manner where the tracking outputs are fed to the forecasting module. **(Bottom)** Our proposed model jointly achieves the tracking and forecasting. Also, we propose two innovations: (1) a feature interaction using GNNs (shown as blue) to improve the tracking association and trajectory forecasting in the presence of multiple agents; (2) a diversity sampling (shown as orange) to improve the sample efficiency and produce diverse and accurate trajectory samples.

To enable the error feedback mechanism in MOT and trajectory forecasting modules, we propose to jointly optimize both modules. Specifically, instead of running two modules separately as shown in Fig. 1 (top), our proposed method performs tracking and forecasting simultaneously as shown in Fig. 1 (bottom). As a result, the error computed in both heads can be propagated back to affect the feature learning for both tasks, which we believe will lead to a better shared feature representation through our joint learning. As trajectory forecasting can implicitly force the network to learn object motion dynamics, we believe that it can lead to more discriminative feature learning for 3D MOT association, which vice versa can improve the downstream trajectory forecasting module.

Modeling interaction is crucial in the presence of multiple agents, which however has been overlooked in prior work, especially in 3D MOT. As shown in Fig. 1 (top), prior work in 3D MOT extracts the feature of each object *independently*, i.e., the feature of an object does not interact with the features of other objects. We found that *this independent feature extraction is sub-optimal for discriminative feature learning*. This is because feature similarity of different objects should be dependent on MOT. For example, if the feature similarity of two objects is

increased, the feature similarity between any one of these two objects and other objects should be decreased to avoid confusion in the MOT association.

To model object interaction in 3D MOT, we propose a novel feature interaction mechanism as shown in Fig. 1 (Bottom). We achieve this by introducing the Graph Neural Networks (GNNs) to 3D MOT. To the best of our knowledge, our work is the first to apply GNNs to MOT. Specifically, we construct a graph with each node being an object in the scene. Then, at every layer of the GNNs, each node can update its feature by aggregating features from other nodes. This node feature aggregation process is useful because the resulting object features are no longer isolated and are adapted according to other objects. We observe in our experiments that, after a few GNN layers, the affinity matrix becomes more discriminative than the affinity matrix obtained without feature interaction. In addition to employing GNNs for 3D MOT, GNNs feature interaction is also shared with the trajectory forecasting head. Although a few works [38,33] have used GNNs for trajectory forecasting, we are the first to employ GNNs in an unified 3D MOT and trajectory forecasting method.

As the object future trajectories are stochastic and multi-modal due to unobserved factors (e.g., intentions), prior work in trajectory forecasting often learns the future trajectory distribution with deep generative models such as conditional variational autoencoders (CVAEs; [41]) and conditional generative networks (CGANs; [41]). At test time, these methods randomly sample a set of future trajectories from the generative model without considering the correlation between samples. As a result, the samples can be very similar and only cover a limited number of modes, leading to poor sample efficiency. This inefficient sampling technique is harmful in real-time applications because producing a large number of samples can be computationally expensive and lead to high latency. Moreover, without covering all the modes in the trajectory distribution and considering all possible futures, the perception system cannot plan safely, which is important in safety-critical applications such as autonomous driving.

To improve sample efficiency in trajectory forecasting, we depart from the random sampling in prior work and employ a diversity sampling technique that can generate accurate and diverse trajectory samples from a pretrained CVAE model. The idea is to learn a separate sampling network which maps the object feature to a set of latent codes. The latent codes are then decoded into trajectory samples. In this way, the produced samples are correlated (unlike random sampling where the samples are independent), which allows us to enforce structural constraints such as diversity onto the samples. Specifically, we use determinantal point processes (DPPs; [40]) to optimize the diversity of the samples.

Our contributions are summarized as follows:

1. **A joint 3D MOT and trajectory forecasting model**, improving the performance of both modules through joint optimization;
2. **A novel feature interaction mechanism using GNNs**. To the best of our knowledge, we are the first to introduce GNNs for 3D MOT.
3. **Introducing diversity sampling for multi-agent trajectory forecasting**, that can produce more accurate and diverse trajectory samples.

2 Related Work

3D Multi-Object Tracking. Recent work approaches 3D MOT in an online fashion using a tracking-by-detection pipeline, where the performance is mostly affected by two factors: 3D detection quality and discriminative feature learning. To obtain the discriminative feature, prior work focuses on feature engineering, among which the motion and appearance features are the most popular. [4,19,31] employ Convolutional Neural Networks (CNNs) to extract 2D appearance feature. To learn the 3D appearance feature from a point cloud, [74] proposes a PointNet [15] based 3D MOT network. To leverage motion features, filter-based methods [61,67] and learning-based methods [4] have been proposed. Although prior work has achieved impressive performance by feature engineering, they perform feature extraction for each object independently, ignoring interaction between the objects. To the best of our knowledge, our work is the first attempt to improve discriminative feature learning by introducing GNNs to 3D MOT.

Trajectory Forecasting is to predict a sequence of ground positions of target objects in the future. Prior work mostly investigates the target object of people [37,1,55,27,38,56,33,71] and vehicles [41,54,53,10,17,44]. As the future is multi-modal, prior works [27,33,38,25] employ probabilistic models for trajectory prediction. Also, because agent behavior is strongly influenced by their with other agents, [38,33,44] introduce GNNs to learn the interaction-aware feature for trajectory forecasting. However, most prior works study trajectory forecasting separately from the highly-related tracking module, while we consider both forecasting and tracking in a unified framework.

Joint 3D Detection, Tracking and Forecasting. On a few of the prior works attempt joint optimization for different combinations of the three modules. [31,19,9,61] learn a joint 3D detector and tracker. The resulting past object trajectories can be fed to a separate trajectory forecasting module for prediction. [72,8] achieve the joint detection and trajectory forecasting, where the detected boxes are directly used for trajectory prediction, skipping the tracking association. To the best of our knowledge, no existing work has investigated joint optimization for tracking and forecasting as in our proposed method. Perhaps [47] is the most related work which proposes a method to perform detection, tracking and forecasting. However, similar to [72,8], only detection and forecasting are jointly optimized during training. The tracklet outputs are then obtained with a separate post-processing step during testing. Different from [47], our method jointly optimizes both tracking and forecasting modules during training and verifies that the joint optimization improves the performance of both modules.

Graph Neural Networks was first proposed by [24] to directly process graph-structured data using neural networks. The major component of GNNs is a node feature aggregation technique, with which the node feature can be updated by interacting with other nodes. With this feature interaction technique, significant success has been achieved by introducing GNNs to vision applications such as semantic segmentation [14,73], action recognition [66,43,59,75], single object tracking [20], person re-identification [69]. However, no existing work has introduced GNNs to 3D MOT yet. To the best of our knowledge, our work is the

first attempt using GNNs to model interaction in 3D MOT, and also the first to introduce GNNs for joint 3D MOT and trajectory forecasting.

Diversity Sampling. Stemming from the M-Best MAP problem [51,58], diverse M-Best solutions [5] and multiple choice learning [28,42] are able to produce a diverse ensemble of solutions and models. Also, submodular function maximization [30] has been used for diverse selection of garments from fashion images. Determinantal point processes (DPPs) [48,40] are also popular probabilistic models for subset selection due to its ability to measure the global diversity and quality within a set. Prior work has applied DPPs for document and video summarization [39,23], recommendation systems [22], object detection [3], and grasp clustering [32]. Sample diversity has also been an active research topic in generative modeling. A majority of this line of research aims to improve the diversity of the data distribution learned by deep generative models, including works that try to alleviate the mode collapse problem in GANs [12,13,62,2,26,18,45] and the posterior collapse problem in VAEs [76,63,35,46,29]. Recent work [70] has employed DPPs to improve sample diversity in single-agent trajectory prediction and evaluated the performance on a toy dataset. Different from prior work, we are the first to apply the diversity sampling technique to multi-agent trajectory forecasting and evaluate on real-world large-scale datasets.

3 Approach

The aim of the proposed method is to achieve 3D MOT and trajectory forecasting simultaneously. Let $\mathcal{O} = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$ denote the set of past trajectories of M tracked objects. Each past trajectory $\mathbf{o}_i = [\mathbf{o}_i^{-H}, \dots, \mathbf{o}_i^{-1}]$ consists of the associated detections of the i -th tracked object in the past H frames. The detection at frame $t \in \{-H, \dots, -1\}$ is a tuple $\mathbf{o}_i^t = [x, y, z, l, w, h, \theta, I]$, where (x, y, z) denotes the object center in 3D space, (l, w, h) denotes the object size, θ is the heading angle, and I is the assigned ID. Let $\mathcal{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_N\}$ denote the set of unassociated detections of N objects in the current frame obtained by a 3D object detector. Each unassociated detection $\mathbf{d}_j = [x, y, z, l, w, h, \theta]$ is defined similarly to the past associated detections \mathbf{o}_i^t except without the assigned ID I . The goal of 3D MOT is to associate the current detection $\mathbf{d}_j \in \mathcal{D}$ with the past object trajectory $\mathbf{o}_i \in \mathcal{O}$ and assign an ID to \mathbf{d}_j . For trajectory forecasting, the objective is to predict the future trajectories $\mathcal{F} = \{\mathbf{f}_i, \dots, \mathbf{f}_M\}$ for all M tracked objects in the past. Each future trajectory $\mathbf{f}_i = [\mathbf{f}_i^1, \dots, \mathbf{f}_i^T]$ consists of the x and z positions (i.e., 2D position ground in a top-down view) of i -th object in future T frames, i.e., $\mathbf{f}_i^t = [x, z]$ where $t \in \{1, \dots, T\}$.

The entire network of our method to achieve the joint tracking and forecasting is shown in Fig. 1 (bottom), which consists of five modules: (1) a feature extractor to encode the feature for the object trajectories in the past and the detections in the current frame; (2) a feature interaction mechanism using GNNs to update the object features based on the features of other objects; (3) a 3D MOT head that computes the affinity matrix for data association between the tracked objects in the past and detected objects in the current frame; (4) a trajectory forecasting head that learns a CVAE to generate future trajectories based on the GNN

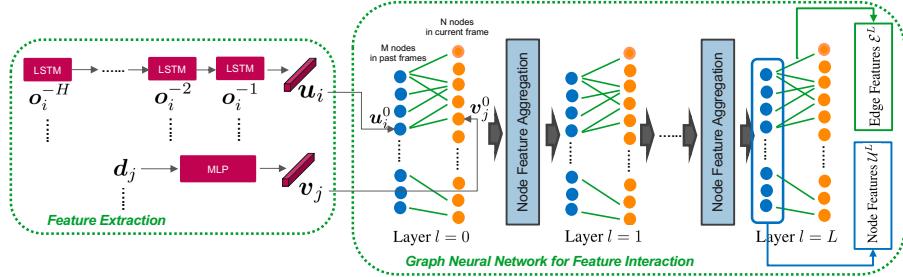


Fig. 2: (Left) To leverage the location and motion cues, we extract the feature from object trajectories $\mathcal{O} = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$ in the past using an LSTM model and extract the feature from detections $\mathcal{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_N\}$ in the current frame using a MLP. **(Right)** The proposed feature interaction mechanism based on GNN is used to update the object node feature $\mathcal{U}^l = \{\mathbf{u}_1^l, \dots, \mathbf{u}_M^l\}$ and $\mathcal{V}^l = \{\mathbf{v}_1^l, \dots, \mathbf{v}_N^l\}$ at GNN layer l and iteratively through all GNN layers. At the final layer, we use the node features for tracked objects \mathcal{U}^L for 3D MOT task (see Sec. 3.3), and use the edge features \mathcal{E}^L (computed from \mathcal{U}^L and \mathcal{V}^L) for trajectory forecasting task (see Sec. 3.4 and 3.5).

features and past trajectories; (5) a diversity sampling that can optimize the diversity of the trajectory samples.

3.1 Feature Extraction

To utilize motion and location information from the object trajectories in the past and detections in the current frame for tracking and forecasting, we first learn feature extractors to encode the information as shown in Fig. 2 (Left). Given a trajectory $\mathbf{o}_i = [\mathbf{o}_i^{-H}, \dots, \mathbf{o}_i^{-1}]$ for tracked object i , we obtain its object feature by applying a two-layer LSTM to model the temporal dynamics in the data and output a feature \mathbf{u}_i with 64 dimensions. For a detected object j in the current frame, we use a 2-layer Multi-Layer Perceptron (MLP) to map the detection \mathbf{d}_j to a 64-dimensional feature \mathbf{v}_j . Note that the feature extractors for tracked object \mathbf{o}_i and detected object \mathbf{d}_j are different as \mathbf{o}_i and \mathbf{d}_j have different time horizon. The obtained features \mathbf{u}_i and \mathbf{v}_j for tracked and detected objects are then used as the initial node features \mathbf{u}_i^0 and \mathbf{v}_j^0 at layer 0 of the following GNN module (please see Sec. 3.2) for feature interaction.

Note that our feature extractor is shared and optimized for both tracking and trajectory forecasting, which is different from prior work that applies the feature extraction twice separately in 3D MOT and trajectory forecasting as shown in Fig. 1 (top). As a result, our proposed joint model removes redundant feature extraction part, which reduces the complexity of the entire system.

3.2 Graph Neural Network for Feature Interaction

Graph Construction. After feature extraction, we should have obtained M features $\{\mathbf{u}_1^0, \dots, \mathbf{u}_M^0\}$ for tracked objects in the past and N features $\{\mathbf{v}_1^0, \dots, \mathbf{v}_N^0\}$ for detected objects in the current frame. We then construct an L -layer Graph Neural Network (GNN) where each layer includes the nodes of the M tracked objects and N currently detected objects (shown in Fig. 2 (right)). As the node

feature will be updated at each layer of the GNN, let us denote the node features for the tracked objects at layer l as $\mathcal{U}^l = \{\mathbf{u}_1^l, \dots, \mathbf{u}_M^l\}$. Similarly, we define the node feature for the currently detected objects at layer l as $\mathcal{V}^l = \{\mathbf{v}_1^l, \dots, \mathbf{v}_N^l\}$.

In addition to node feature definition, we also define the set of edge features in the graph as $\mathcal{E}^l = \{\mathbf{e}_{ij}^l, \dots, \mathbf{e}_{ij}^l\}$, where \mathbf{e}_{ij}^l is an edge feature relating node feature \mathbf{u}_i^l and \mathbf{v}_j^l at layer l of the graph. To make the GNN learning efficient, we restrict the edge connection to be sparse, i.e., the edge feature is not defined between all pairs of nodes. Specifically, we utilize the prior knowledge of MOT and trajectory forecasting: (1) the objects' interaction should only happen for objects that are close to each other (e.g., objects in the front and behind, or objects that are crossing in an intersection); (2) the MOT association should only happen between tracked objects and detected objects. Therefore, we construct the edge for a pair of nodes if and only if (1) two nodes' box centers have distance less than 15 meters in 3D space and (2) one node is for a past tracked object and the other node belongs to a currently detected object. As a result, we have a sparse edge connection in our GNNs as shown in Fig. 2 (right).

Node Feature Aggregation. To model node feature interaction in GNN, we iteratively update the node feature by aggregating features from the neighborhood nodes (i.e., nodes connected by an edge) in each layer. Specifically, we employ the node feature aggregation rule proposed in GraphConv [50]:

$$\mathbf{u}_i^{l+1} = \sigma_1^l(\mathbf{u}_i^l) + \sum_{j \in \mathcal{N}(i)} \sigma_2^l(\mathbf{v}_j^l), \quad (1)$$

where \mathbf{u}_i^l and \mathbf{u}_i^{l+1} are the node features for tracked objects at layer l and $l+1$. $\mathcal{N}(i)$ denotes a set of neighborhood nodes for detected objects that are connected to the node i by an edge, and \mathbf{v}_j^l with $j \in \mathcal{N}(i)$ is an neighborhood node feature at layer l . Moreover, σ_1^l , σ_2^l are linear layers at layer l , and the weights of these linear layers are not shared across layers. Note that a ReLU operator is applied to the node feature after feature aggregation at each layer except for the final layer. Intuitively, the above node aggregation rule means that each node feature is updated by aggregating the transformed features of its own and its connected nodes. In addition to updating the node feature \mathbf{u}_i^l for tracked objects, we also update the node feature \mathbf{v}_j^l for detected objects:

$$\mathbf{v}_j^{l+1} = \sigma_1^l(\mathbf{v}_j^l) + \sum_{i \in \mathcal{N}(j)} \sigma_2^l(\mathbf{u}_i^l). \quad (2)$$

Based on the above rules, the updated node features for tracked objects \mathcal{U}^{l+1} and for detected objects \mathcal{V}^{l+1} will affect each other through feature interaction in the following layers. After several layers of GNN feature interaction, we use the node features for tracked objects \mathcal{U}^L for trajectory forecasting. Due to the feature interaction, we believe that the final node features for tracked objects \mathcal{U}^L should have contained enough information about detected objects in the current frame. We do not use the node features \mathcal{V}^L for trajectory forecasting because \mathcal{V}^L might contain the features of new objects which do not have past trajectories.

Edge Feature. As each entry of the affinity matrix in MOT usually represents similarity of the object features, it is natural to use the edge feature relating two

object node features to compute the affinity matrix. To learn similarity in the edge feature, we first define the edge feature between a pair of connected nodes as the difference of their node features:

$$\mathbf{e}_{ij}^l = \mathbf{u}_i^l - \mathbf{v}_j^l, \quad (3)$$

where \mathbf{u}_i^l and \mathbf{v}_j^l are two node features related by the edge feature \mathbf{e}_{ij}^l at the layer l of GNN. After several layers of feature interaction, we use the edge features \mathcal{E}^L at the final GNN layer for 3D MOT association.

3.3 3D Multi-Object Tracking Head

To solve 3D MOT association, we need to learn an affinity matrix A based on pairwise similarity of the features extracted from M tracked objects in the past and N detected objects in the current frame. As a result, affinity matrix A has a dimension of $M \times N$ where each entry A_{ij} represents the similarity score between the tracked object i and the detected object j .

Edge Regression. To learn the affinity matrix A , we employ an edge regression module as shown in Fig. 3, which consists of a two-layer MLP with a non-linear operator and a Sigmoid layer. To compute each entry A_{ij} , the edge regression module uses the edge feature \mathbf{e}_{ij}^L as input and outputs a scalar value between 0 to 1 as the pairwise similarity score:

$$A_{ij} = \text{Sigmoid}(\sigma_4(\text{ReLU}(\sigma_3(\mathbf{e}_{ij}^L)))), \quad (4)$$

where σ_3 and σ_4 are two linear layers. As a result, the computed affinity matrix A can be used to associate the objects using the Hungarian algorithm [64] during testing time. For tracked objects and detected objects that cannot be associated, we employ the same birth and death memory as in [67] to create and delete identities. During training, we learn the network parameters by computing an affinity loss between the estimated affinity matrix A and its ground truth (GT).

Affinity Loss. As shown in Fig. 3, we employ an affinity loss \mathcal{L}_{aff} to directly supervise the output A of 3D MOT head. Our affinity loss consists of two individual losses. First, as we know that the GT affinity matrix A^g can only have integer 0 or 1 on all the entries, we can formulate the prediction of the affinity matrix as a binary classification problem. Therefore, our first loss is the binary cross entropy loss \mathcal{L}_{bce} that is applied on each entry of A :

$$\mathcal{L}_{\text{bce}} = \frac{-1}{MN} \sum_{i=1}^M \sum_{j=1}^N A_{ij}^g \log A_{ij} + (1 - A_{ij}^g) \log(1 - A_{ij}). \quad (5)$$

Second, we know that each tracked object \mathbf{o}_i can only have either one matched detection \mathbf{d}_j or no match at all. In other words, each row and column of the A^g can only be a one-hot vector or an all-zero vector. This motivates our second

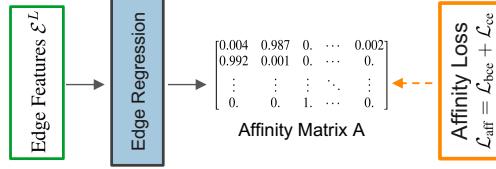


Fig. 3: 3D MOT head with an affinity loss.

loss. For all rows and columns that have a one-hot vector in A^g , we apply the cross entropy loss \mathcal{L}_{ce} to the corresponding rows and columns of A . As an example, the j th column $A_{:,j}^g$ in GT affinity matrix is a one-hot vector and the loss \mathcal{L}_{ce} for the j th column is defined as:

$$\mathcal{L}_{\text{ce}} = -\frac{1}{M} \sum_{i=1}^M A_{ij}^g \log \left(\frac{\exp A_{ij}}{\sum_{i=1}^M \exp A_{ij}} \right). \quad (6)$$

We can now summarize the affinity loss \mathcal{L}_{aff} for the 3D MOT head:

$$\mathcal{L}_{\text{aff}} = \mathcal{L}_{\text{bce}} + \mathcal{L}_{\text{ce}}. \quad (7)$$

3.4 Trajectory Forecasting Head

Our trajectory forecasting head aims to learn a conditional generative model $p_\theta(\mathbf{f}_i | \mathbf{o}_i, \mathbf{u}_i^L)$, which learns the distribution of the i -th tracked object's future trajectory \mathbf{f}_i based on its past trajectory \mathbf{o}_i and the corresponding node feature \mathbf{u}_i^L at the last GNN layer. As we share the generative model for all tracked objects \mathcal{O} , we drop the subscripts and superscripts for ease of notation and denote the generative model as $p_\theta(\mathbf{f} | \mathbf{o}, \mathbf{u})$. We adopt the CVAE [41] as our generative model and introduce a latent variable \mathbf{z} to model unobserved factors (e.g., agent intentions) and capture the multi-modal distribution of the future trajectory \mathbf{f} . Based on the CVAE formulation, we introduce a variational lower bound $\mathcal{V}_{lb}(\mathbf{f}; \theta, \phi)$ of the log-likelihood function $\log p_\theta(\mathbf{f} | \mathbf{o}, \mathbf{u})$:

$$\mathcal{V}_{lb}(\mathbf{f}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z} | \mathbf{f}, \mathbf{o}, \mathbf{u})} [\log p_\theta(\mathbf{f} | \mathbf{z}, \mathbf{o}, \mathbf{u})] - \text{KL}(q_\phi(\mathbf{z} | \mathbf{f}, \mathbf{o}, \mathbf{u}) \| p(\mathbf{z})), \quad (8)$$

where $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a Gaussian latent prior, $q_\phi(\mathbf{z} | \mathbf{f}, \mathbf{o}, \mathbf{u}) = \mathcal{N}(\boldsymbol{\mu}, \text{Diag}(\boldsymbol{\sigma}^2))$ is an approximated posterior (encoder distribution) and $p_\theta(\mathbf{f} | \mathbf{z}, \mathbf{o}, \mathbf{u}) = \mathcal{N}(\tilde{\mathbf{f}}, \alpha \mathbf{I})$ is a conditional likelihood (decoder distribution) with a coefficient α . We use two Recurrent Neural Networks as the encoder F_ϕ and decoder G_θ to respectively output the parameters of the encoder and decoder distributions: $(\boldsymbol{\mu}, \boldsymbol{\sigma}) = F_\phi(\mathbf{f}, \mathbf{o}, \mathbf{u})$ and $\tilde{\mathbf{f}} = G_\theta(\mathbf{z}, \mathbf{o}, \mathbf{u})$. The detailed architectures for F_ϕ and G_θ are given in Appendix A. Based on above formulation, the loss for our trajectory forecasting head is $\mathcal{L}_{\text{cvae}} = -\mathcal{V}_{lb}$. As we jointly optimize the tracking and forecasting heads as well as the feature extractors and GNNs, we summarize the overall loss of our network as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{aff}} + \mathcal{L}_{\text{cvae}}. \quad (9)$$

Once the CVAE model is learned during training, we can produce i -th agent's future trajectories \mathbf{f}_i by randomly sampling a set of latent codes $\{\mathbf{z}_{i1}, \dots, \mathbf{z}_{iK}\}$ from the latent prior and decode them using the decoder G_θ into future trajectory samples $\{\mathbf{f}_{i1}, \dots, \mathbf{f}_{iK}\}$. However, as random sampling can lead to similar samples and low sample efficiency, we introduce a diversity sampling technique into multi-agent trajectory forecasting that can produce both diverse and accurate samples, and improve the sample efficiency.

3.5 Diversity Sampling Technique

To obtain diverse future trajectory samples from the pretrained CVAE model, we introduce the diversity sampling technique to our multi-agent trajectory forecasting. As shown in Fig. 4, we use a γ -parameterized Diversity Sampling Function

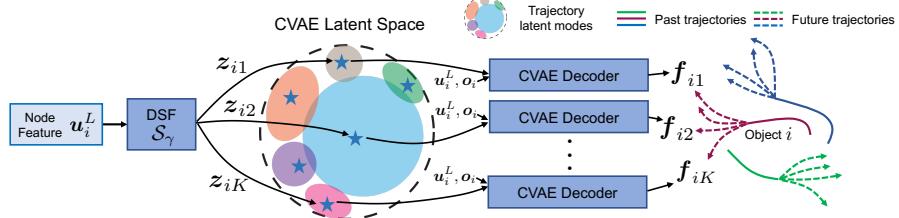


Fig. 4: Trajectory Forecasting with Diversity Sampling. To produce diverse trajectory samples from our pretrained CVAE model, we learn a diversity sampling function (DSF) \mathcal{S}_γ to map each object’s node feature \mathbf{u}_i^L to a set of latent codes, which can cover not only the major mode but also other modes in the CVAE latent space. Then, we decode those codes into diverse and accurate future trajectories for object i .

(DSF) \mathcal{S}_γ (a two-layer MLP) that maps the i -th object’s GNN feature \mathbf{u}_i^L to a set of latent codes: $\mathcal{S}_\gamma(\mathbf{u}_i^L) = \{\mathbf{z}_{i1}, \dots, \mathbf{z}_{iK}\}$. We can then use the CVAE decoder G_θ to decode the latent codes into a set of future trajectories $\mathcal{Y}_i = \{\mathbf{f}_{i1}, \dots, \mathbf{f}_{iK}\}$ for object i . In this way, the latent codes and trajectory samples are correlated and can be controlled by the parameters of the DSF \mathcal{S}_γ . Our goal is to optimize \mathcal{S}_γ so that the trajectory samples \mathcal{Y}_i are both diverse and accurate. To model diversity, we construct a DPP kernel $\mathbf{L}^i \in \mathbb{R}^{K \times K}$ for each object i based on the diversity and quality of the samples in \mathcal{Y}_i :

$$\mathbf{L}^i = \text{Diag}(\mathbf{r}^i) \cdot \mathbf{S}^i \cdot \text{Diag}(\mathbf{r}^i), \quad (10)$$

where the DPP kernel is formed by two components representing sample diversity and quality — a similarity matrix $\mathbf{S}^i \in \mathbb{R}^{K \times K}$ and a quality vector $\mathbf{r}^i \in \mathbb{R}^K$:

$$\mathbf{S}_{ab}^i = \exp(-\omega \|\mathbf{f}_{ia} - \mathbf{f}_{ib}\|^2), \quad \mathbf{r}_k^i = \exp(\max(-\|\mathbf{z}_{ik}\|^2 + R^2, 0)). \quad (11)$$

The element \mathbf{S}_{ab}^i of the similarity matrix measures the similarity between trajectory samples \mathbf{f}_{ia} and \mathbf{f}_{ib} with a Gaussian kernel where ω is a scaling factor. Each element \mathbf{r}_a^i in the quality vector defines the quality of sample \mathbf{f}_{ia} based on how far its latent code \mathbf{z}_{ik} is from the origin. If \mathbf{z}_{ik} is very far, it means the sample has low likelihood and will be assigned a low quality score. Based on the DPP kernel \mathbf{L}^i , one can define a diversity loss to measure the diversity and quality within the trajectory samples \mathcal{Y}_i :

$$\mathcal{L}_{\text{dpp}}^i = \sum_{n=1}^N \frac{\lambda_n}{\lambda_n + 1} = -\text{tr}(\mathbf{I} - (\mathbf{L}^i + \mathbf{I})^{-1}), \quad (12)$$

where λ_n is the n -th eigenvalue of \mathbf{L}^i , $\text{tr}(\cdot)$ is the trace operator and \mathbf{I} is the identity matrix. As the diagonal elements of \mathbf{L}^i are all ones, the sum of eigenvalues is fixed: $\sum \lambda_n = \text{tr}(\mathbf{L}^i) = K$. The optima of $\mathcal{L}_{\text{dpp}}^i$ is obtained when all eigenvalues are equal and \mathbf{L}^i becomes an identity matrix, thus making $\mathbf{S}_{ab}^i = 0$ ($a \neq b$) and $\mathbf{r}_k^i = 1$. This means the distance between trajectory samples is large and each sample has high likelihood. However, this optima is seldom obtained due to the trade-off between diversity and quality, i.e., samples far away from others often have low likelihood. Besides the diversity loss, we further introduce

a reconstruction loss to encourage the set of trajectory samples \mathcal{Y}_i to cover the ground-truth future trajectories $\hat{\mathbf{f}}$:

$$\mathcal{L}_{\text{recon}}^i = \min_k \|\mathbf{f}_{ik} - \hat{\mathbf{f}}\|^2. \quad (13)$$

To learn the DSF \mathcal{S}_γ , we freeze the parameters all other components in our framework (feature extractors, GNNs, and CVAE), and only optimize the parameters γ of the DSF with the following loss:

$$\mathcal{L}_{\text{dsf}} = \frac{1}{M} \sum_{i=1}^M \mathcal{L}_{\text{dpp}}^i + \mathcal{L}_{\text{recon}}^i. \quad (14)$$

4 Experiments

4.1 Settings

Dataset. To evaluate our method for joint 3D MOT and trajectory forecasting, we use standard 3D MOT datasets including KITTI-tracking [21] and nuScenes-tracking [7]. Both datasets provide 3D detection and trajectory GT for evaluation. Same as prior work that focuses the experiments on predicting trajectories of either people [1,55,27,38,56,33,37] or vehicles [41,54,53,10,17,44] despite that most methods are generic to both types of objects, we restrict the scope of our experiments for vehicles only. Specifically, we use the car subset in both datasets. Also, since there is no existing evaluation procedure that can jointly evaluate 3D MOT and trajectory forecasting, we evaluate two modules separately and compare with prior work on each individual module.

4.2 Evaluating 3D Multi-Object Tracking

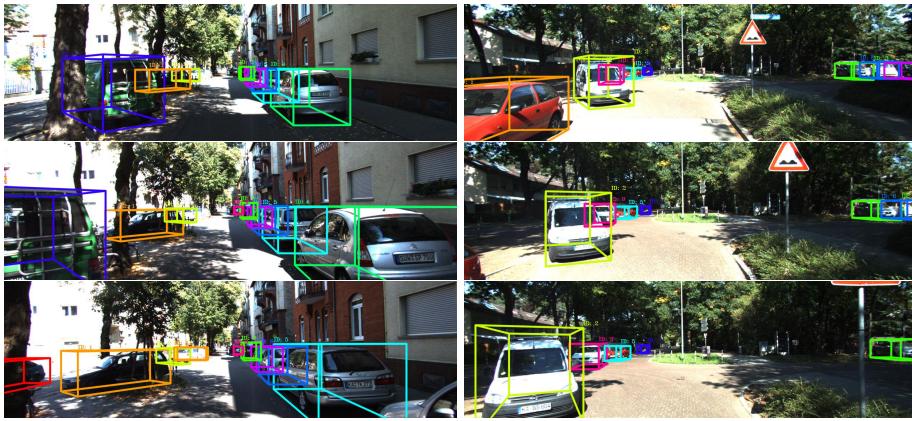
Evaluation Metrics. We use standard CLEAR metrics [6] (including MOTA, MOTP, IDS) and also the new sAMOTA, AMOTA and AMOTP metrics proposed in [67] for evaluation. Additionally, for test videos, KITTI and nuScenes datasets do not release the ground truth to users but reserve it on the evaluation server. Therefore, to evaluate 3D MOT systems using the above metrics, some of which are not included in the evaluation server, we use the validation set for evaluation. Note that, Although nuScenes server evaluates the AMOTA, it is implemented different from the sAMOTA/AMOTA evaluation in the code released by [67]. We only report the number using the code from [67].

Baselines. We compare against recent open-source 3D MOT systems such as FANTrack [4], mmMOT [74], 3DT [31] and AB3DMOT [67]. We use the same 3D detections obtained by PointRCNN [60] on KITTI and by Megvii [77] on nuScenes for our proposed method and also the 3D MOT baselines [4,74,67] that require 3D detections as inputs. For the baselines [31,4] that also require 2D detections as inputs, we use 2D projection of the 3D detections.

Results. We summarize the 3D MOT results on KITTI and nuScenes datasets in Table 1. Our method consistently outperforms baselines in sAMOTA, AMOTA and MOTA, which are the primary metrics for ranking MOT methods. We hypothesize that this is because our method leveraging GNN obtains more discriminative features to avoid confusion in MOT association while all 3D MOT

Table 1: 3D MOT Evaluation on the KITTI and nuScenes datasets.

Datasets	Methods	sAMOTA(%)↑	AMOTA(%)↑	AMOTP(%)↑	MOTA(%)↑	MOTP(%)↑	IDS↓
KITTI	3DT [31]	59.52	27.92	63.12	59.82	64.45	228
	FANTrack [4]	82.97	40.03	75.01	74.30	75.24	35
	mmMOT [74]	86.39	40.55	73.24	79.02	75.07	30
	AB3DMOT[67]	91.78	44.26	77.41	83.35	78.43	0
	Ours	92.37	44.96	76.83	84.49	78.32	3
nuScenes	3DT [31]	6.55	-0.41	15.61	12.08	34.75	794
	FANTrack [4]	19.64	2.36	22.92	18.60	39.82	1593
	mmMOT [74]	23.93	2.11	21.28	19.82	40.93	572
	AB3DMOT[67]	27.90	4.93	23.89	21.46	41.02	395
	Ours	28.96	11.36	25.83	22.81	41.99	451

**Fig. 5:** 3D MOT visualization on two sequences of the KITTI dataset.

baselines ignore the interaction between objects. Moreover, joint optimization of the tracking and forecasting modules also helps. We will justify both hypotheses in the ablation study. We show qualitative results of our method on the KITTI dataset in Fig. 5, demonstrating reliable 3D MOT performance.

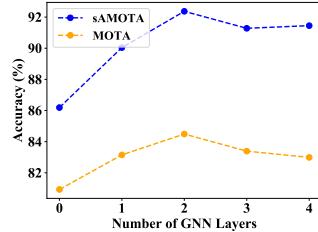
Ablation Study. We first verify if the joint tracking and forecasting improves the performance of the tracking. In Table 2, when we train the MOT and trajectory forecasting heads together on the KITTI dataset, the performance is higher in most metrics compared to the model without the forecasting. This proves that unifying both tasks is beneficial to 3D MOT. Secondly, as shown in Fig. 6, we validate on the KITTI dataset that the feature interaction using GNNs is helpful for 3D MOT association. We can see that, the performance is significantly increased with the first two layers of GNN, and then starts to converge with more than two layers. As a result, we use two GNN layers in the best model.

4.3 Evaluating the Trajectory Forecasting

Evaluation Metrics. We use the standard metrics: Average Displacement Error (ADE) [1] and Final Displacement Error (FDE) for accuracy measurement. Additionally, to evaluate the diversity of the trajectory samples and penalize similar samples, we use the Average Self Distance (ASD) and Final Self Distance (FSD) metrics proposed in [70] for sample diversity evaluation.

Table 2: Effect of trajectory forecasting heading on 3D MOT.

Metrics	w/o forecasting	w/ forecasting
sAMOTA(%)↑	90.17	92.37
AMOTA(%)↑	42.81	44.96
AMOTP(%)↑	76.94	76.83
MOTA(%)↑	82.91	84.49
MOTP(%)↑	78.11	78.32
IDS↓	5	3

**Fig. 6:** Effect of GNNs on 3D MOT.**Table 3:** Trajectory forecasting evaluation on the KITTI and nuScenes datasets.

Datasets	Metrics	Conv-Social [17]	Social-GAN [27]	TraPHic [10]	Graph-LSTM [11]	Ours
KITTI-1.0s	ADE↓	0.607	0.586	0.542	0.478	0.471
	FDE↓	0.948	1.167	0.839	0.800	0.763
	ASD↑	1.785	0.495	1.787	1.070	2.351
	FSD↑	1.987	0.844	1.988	1.836	4.071
KITTI-3.0s	ADE↓	2.362	2.340	2.279	1.994	1.319
	FDE↓	3.916	4.102	3.780	3.351	2.299
	ASD↑	2.436	1.351	2.434	2.745	5.843
	FSD↑	2.973	2.066	2.973	4.582	10.123
nuScenes-1.0s	ADE↓	0.674	0.483	0.571	0.509	0.378
	FDE↓	0.784	0.586	0.640	0.618	0.490
	ASD↑	2.101	1.005	2.102	1.122	5.665
	FSD↑	2.430	1.475	2.432	1.603	7.826
nuScenes-3.0s	ADE↓	1.989	1.794	1.827	1.646	1.017
	FDE↓	3.015	2.850	2.760	2.445	1.527
	ASD↑	2.799	1.945	2.803	2.742	8.323
	FSD↑	4.174	3.610	4.184	4.970	15.787

Baselines. As our focus is to forecast trajectories for vehicles, we mostly compare against methods designed for vehicle trajectory prediction [17,10,11], among which [11] also leverages GNNs. Additionally, we also compare with the pioneering work in multi-agent trajectory forecasting literature: Social-GAN [27]. As Social-GAN is originally designed for pedestrian trajectory prediction, we adapt it to predict trajectories of vehicles. Following prior work [27], we use 20 samples for all methods. Also, to compare with our method which predicts the future trajectories in frames $t \in \{1, \dots, T\}$ based on GT past trajectories in frames $t \in \{-H, \dots, -1\}$ and detection in frame 0, we experiment with the trajectory forecasting baselines (which do not have a joint MOT module) to predict future trajectories in frames $t \in \{1, \dots, T\}$ based on past frames $t \in \{-H, \dots, 0\}$ so that the baselines do not miss the information at the frame 0.

Results. We summarize the trajectory forecasting results on the KITTI and nuScenes datasets in Table 3. Our proposed trajectory forecasting method, which (1) is jointly trained with a 3D MOT head, (2) uses GNNs for feature interaction and (3) uses a diversity sampling, outperforms the baselines in both accuracy and diversity metrics. Particularly, our method outperforms baselines by a large margin for the long-horizon (i.e., 3.0s) experiment. This is because our method has a higher sample efficiency and can cover different modes of the future trajectory distribution. We show qualitative results of our method on the KITTI dataset in Fig. 7 with plausible and diverse trajectory predictions.

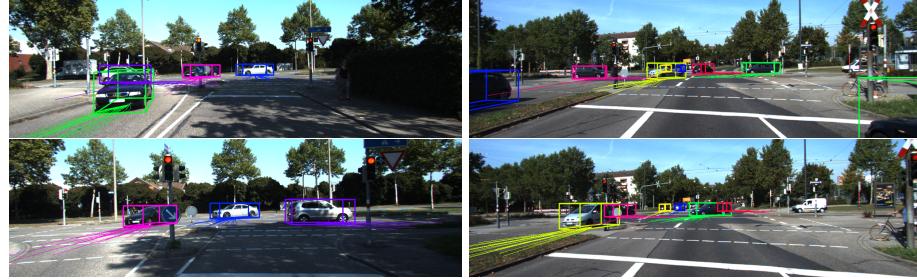


Fig. 7: Trajectory forecasting visualization on the KITTI dataset.

Table 4: Effect of 3D MOT and diversity sampling function on trajectory forecasting.

Datasets	Metrics	w/o MOT+DSF	w/o DSF	Ours
KITTI-1.0s	ADE \downarrow	0.663	0.582	0.471
	FDE \downarrow	1.121	0.978	0.763
	ASD \uparrow	1.796	1.730	2.351
	FSD \uparrow	3.168	3.052	4.071
KITTI-3.0s	ADE \downarrow	1.729	1.564	1.319
	FDE \downarrow	3.086	2.893	2.299
	ASD \uparrow	3.196	3.416	5.843
	FSD \uparrow	5.776	6.168	10.123

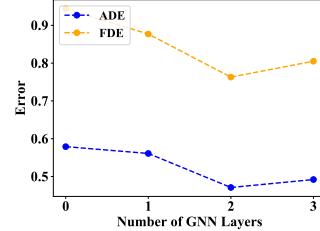


Fig. 8: Effect of GNN feature interaction on trajectory forecasting.

Ablation Study. We verify if the joint tracking/forecasting optimization and diversity sampling function improves the performance of the trajectory forecasting module on the KITTI dataset. In Table 4, we denote our model trained without the MOT head and DSF as w/o MOT+DSF. Then, we add one module at a time. We first add the MOT head, denoted as w/o DSF in Table 4, showing significant improvement in accuracy metrics and slight improvement in diversity metrics. We believe it is because that the auxiliary tracking objective improves the shared feature learning, which is helpful to trajectory forecasting. Moreover, after adding the DSF module, denoted as **Ours** in Table 4, we see further improvement in both the accuracy and diversity metrics. In Fig. 8, we also verify if the feature interaction is helpful for trajectory forecasting. One can observe that the accuracy metrics are increased and reach the highest performance with two layers of GNN, showing the effectiveness of GNN feature interaction.

5 Conclusion

We proposed a unified 3D MOT and trajectory forecasting method and demonstrated that it is beneficial to achieve both tasks under one unified framework through shared feature learning. Also, we incorporated two novel computational units into our approach: (1) a GNN-based feature interaction mechanism, which is introduced for the first time to 3D MOT to improve the discriminative feature learning; (2) a diversity sampling technique that improves sample efficiency for multi-agent trajectory forecasting. Through experiments, we established new state-of-the-art performance on both 3D MOT and trajectory forecasting, showing that the proposed units and joint optimization are effective in our approach.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social LSTM: Human Trajectory Prediction in Crowded Spaces. CVPR (2016)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein Generative Adversarial Networks. ICML (2017)
3. Azadi, S., Feng, J., Darrell, T.: Learning Detection with Diverse Proposals. CVPR (2017)
4. Baser, E., Balasubramanian, V., Bhattacharyya, P., Czarnecki, K.: FANTrack: 3D Multi-Object Tracking with Feature Association Network. arXiv:1905.02843 (2019)
5. Batra, D., Yadollahpour, P., Guzman-Rivera, A., Shakhnarovich, G.: Diverse m-Best Solutions in Markov Random Fields. ECCV (2012)
6. Bernardin, K., Stiefelhagen, R.: Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. Journal on Image and Video Processing (2008)
7. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liang, V.E., Xu, Q.: nuScenes: A Multimodal Dataset for Autonomous Driving. arXiv:1903.11027 (2019)
8. Casas, S., Luo, W., Urtasun, R.: IntentNet: Learning to Predict Intention from Raw Sensor Data. CoRL (2018)
9. Caselitz, T.: Motion-Based Detection and Tracking in 3D LiDAR Scans. ICRA (2016)
10. Chandra, R., Bhattacharya, U., Bera, A., Manocha, D.: TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions. CVPR (2019)
11. Chandra, R., Guan, T., Panuganti, S., Mittal, T., Bhattacharya, U., Bera, A., Manocha, D.: Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs. arXiv:1912.01118 (2019)
12. Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W.: Mode Regularized Generative Adversarial Networks. ICLR (2017)
13. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. NIPS (2016)
14. Chen, Y., Rohrbach, M., Yan, Z., Yan, S., Feng, J., Kalantidis, Y.: Graph-Based Global Reasoning Networks. CVPR (2019)
15. Cherabier, I., Hane, C., Oswald, M.R., Pollefeys, M.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR (2017)
16. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS (2014)
17. Deo, N., Trivedi, M.M.: Convolutional Social Pooling for Vehicle Trajectory Prediction. CVPRW (2018)
18. Elfeki, M., Couprise, C., Riviere, M., Elhoseiny, M.: GDPP: Learning Diverse Generations Using Determinantal Point Process. ICML (2019)
19. Frossard, D., Urtasun, R.: End-to-End Learning of Multi-Sensor 3D Tracking by Detection. ICRA (2018)
20. Gao, J., Zhang, T., Xu, C.: Graph Convolutional Tracking. CVPR (2019)
21. Geiger, A., Lenz, P., Urtasun, R.: Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. CVPR (2012)
22. Gillenwater, J.A., Kulesza, A., Fox, E., Taskar, B.: Expectation-Maximization for Learning Determinantal Point Processes. NIPS (2014)
23. Gong, B., Chao, W.L., Grauman, K., Sha, F.: Diverse Sequential Subset Selection for Supervised Video Summarization. NIPS (2014)

24. Gori, M., Monfardini, G., Scarselli, F.: A New Model for Learning in Graph Domains. IJCNN (2005)
25. Guan, J., Yuan, Y., Kitani, K.M., Rhinehart, N.: Generative Hybrid Representations for Activity Forecasting with No-Regret Learning. CVPR (2020)
26. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved Training of Wasserstein GANs. NIPS (2017)
27. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. CVPR (2018)
28. Guzman-Rivera, A., Batra, D., Kohli, P.: Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. NIPS (2012)
29. He, J., Spokoyny, D., Neubig, G., Berg-Kirkpatrick, T.: Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. ICLR (2019)
30. Hsiao, W.L., Grauman, K.: Creating Capsule Wardrobes from Fashion Images. CVPR (2018)
31. Hu, H.N., Cai, Q.Z., Wang, D., Lin, J., Sun, M., Krähenbühl, P., Darrell, T., Yu, F.: Joint Monocular 3D Vehicle Detection and Tracking. ICCV (2019)
32. Huang, D.A., Ma, M., Ma, W.C., Kitani, K.M.: How do We Use Our Hands? Discovering a Diverse Set of Common Grasps. CVPR (2015)
33. Ivanovic, B., Pavone, M.: The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs. ICCV (2019)
34. Kayukawa, S., Kitani, K.: BBeep: A Sonic Collision Avoidance System for Blind Travellers and Nearby Pedestrians. CHI (2019)
35. Kim, Y., Wiseman, S., Miller, A.C., Sontag, D., Rush, A.M.: Semi-Amortized Variational Autoencoders. ICML (2018)
36. Kingma, D.P., Ba, J.L.: Adam: A Method for Stochastic Optimization. ICLR (2015)
37. Kitani, K.M., Ziebart, B.D., Bagnell, J.A., Hebert, M.: Activity Forecasting. ECCV (2012)
38. Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, S.H., Savarese, S.: Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks. NeurIPS (2019)
39. Kulesza, A., Taskar, B.: k-DPPs: Fixed-Size Determinantal Point Processes. ICML (2011)
40. Kulesza, A., Taskar, B., et al.: Determinantal Point Processes for Machine Learning. Foundations and Trends in Machine Learning (2012)
41. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. CVPR (2017)
42. Lee, S., Prakash, S.P.S., Cogswell, M., Ranjan, V., Crandall, D., Batra, D.: Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. NIPS (2016)
43. Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., Tian, Q.: Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition. CVPR (2019)
44. Li, X., Ying, X., Chuah, M.C.: GRIP: Graph-Based Interaction-Aware Trajectory Prediction. ITSC (2019)
45. Lin, Z., Khetan, A., Fanti, G., Oh, S.: PacGAN: The Power of Two Samples in Generative Adversarial Networks. NIPS (2018)
46. Liu, X., Gao, J., Celikyilmaz, A., Carin, L., et al.: Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. arXiv:1903.10145 (2019)

47. Luo, W., Yang, B., Urtasun, R.: Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. CVPR (2018)
48. Macchi, O.: The Coincidence Approach to Stochastic Point Processes. Advances in Applied Probability (1975)
49. Manglik, A., Weng, X., Ohn-bar, E., Kitani, K.M.: Future Near-Collision Prediction from Monocular Video: Feasibility, Dataset , and Challenges. IROS (2019)
50. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. AAAI (2019)
51. Nilsson, D.: An Efficient Algorithm for Finding the M Most Probable Configurations in Probabilistic Expert Systems. Statistics and computing (1998)
52. Osep, A., Mehner, W., Mathias, M., Leibe, B.: Combined Image- and World-Space Tracking in Traffic Scenes. ICRA (2017)
53. Rhinehart, N., Kris, M., Vernaza, P.: R2P2: A ReparameteRized Pushforward Policy for Diverse, Precise Generative Path Forecasting. ECCV (2018)
54. Rhinehart, N., McAllister, R., Kitani, K., Levine, S.: PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings. ICCV (2019)
55. Robicquet, A., Sadeghian, A., Alahi, A., Savarese, S.: Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes. ECCV (2016)
56. Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, S.H., Savarese, S.: SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. CVPR (2018)
57. Sekii, T.: Robust, Real-Time 3D Tracking of Multiple Objects with Similar Appearances. CVPR (2016)
58. Seroussi, B., Golmard, J.L.: An Algorithm Directly Finding the K Most Probable Configurations in Bayesian Networks. International Journal of Approximate Reasoning (1994)
59. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Skeleton-Based Action Recognition with Directed Graph Neural Networks. CVPR (2019)
60. Shi, S., Wang, X., Li, H.: PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. CVPR (2019)
61. Simon, M., Amende, K., Kraus, A., Honer, J., Sämann, T., Kaulbersch, H., Milz, S., Gross, H.M.: Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds. CVPRW (2019)
62. Srivastava, A., Valkov, L., Russell, C., Gutmann, M.U., Sutton, C.: VEEGAN: Reducing Mode Collapse in GANs Using Implicit Variational Learning. NIPS (2017)
63. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelkopf, B.: Wasserstein Auto-Encoders. ICLR (2018)
64. W Kuhn, H.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly (1955)
65. Wang, S., Jia, D., Weng, X.: Deep Reinforcement Learning for Autonomous Driving. arXiv:1811.11329 (2018)
66. Wang, X., Gupta, A.: Videos as Space-Time Region Graphs. ECCV (2018)
67. Weng, X., Kitani, K.: A Baseline for 3D Multi-Object Tracking. arXiv:1907.03961 (2019)
68. Weng, X., Kitani, K.: Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud. ICCVW (2019)
69. Wu, J., Yang, Y., Liu, H., Liao, S., Lei, Z., Li, S.Z.: Unsupervised Graph Association for Person Re-identification. ICCV (2019)

70. Yuan, Y., Kitani, K.: Diverse Trajectory Forecasting with Determinantal Point Processes. arXiv:1907.04967 (2019)
71. Yuan, Y., Kitani, K.: Ego-Pose Estimation and Forecasting as Real-Time PD Control. ICCV (2019)
72. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-End Interpretable Neural Motion Planner. CVPR (2019)
73. Zhang, L., Li, X., Arnab, A., Yang, K., Tong, Y., Torr, P.H.S.: Dual Graph Convolutional Network for Semantic Segmentation. BMVC (2019)
74. Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J., Loy, C.C.: Robust Multi-Modality Multi-Object Tracking. ICCV (2019)
75. Zhao, R., Wang, K., Su, H., Ji, Q.: Bayesian Graph Convolution LSTM for Skeleton Based Action Recognition. ICCV (2019)
76. Zhao, S., Song, J., Ermon, S.: InfoVAE: Balancing Learning and Inference in Variational Autoencoders. AAAI (2019)
77. Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-Balanced Grouping and Sampling for Point Cloud 3D Object Detection. CVPR (2019)

A Network Architectures

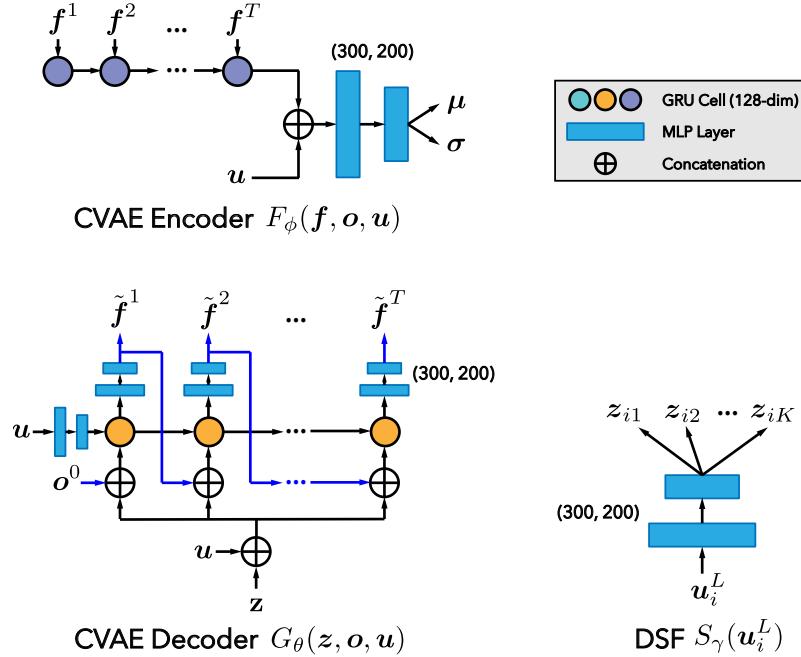


Fig. 9: Network Architectures for the CVAE and diversity sampling function (DSF). We use GRUs [16] to encode and decode future motion. \mathbf{f}^t denotes the position at time t . \mathbf{o}^0 denotes the position at time 0 which is obtained from the tracking module. For the CVAE decoder, we use a 2-layer MLP (300, 200) with Tanh activation to map GNN feature \mathbf{u} to the initial hidden state of the GRU. For all other MLP layers, we use ReLU activation and weights are shared across time steps for the MLP that outputs reconstructed trajectories \mathbf{f} .

B Implementation Details

As mentioned in the main paper, our method follows a two-stage training procedure: we first jointly train the tracking and forecasting modules with the loss $\mathcal{L}_{\text{total}}$ defined in Eq. (9) of the main paper, then we train the diversity sampling function (DSF) while keeping other components of the network fixed. In all experiments, the dimensions of the latent code \mathbf{z} are 32 and the default feature dimensions are 64.

First, we jointly train the tracking and forecasting modules for 50 epochs with Adam [36] and a learning rate of 0.001. We keep the learning rate fixed for 10 epochs and then linearly decay the learning rate to 0. The weight for each loss is 1 (i.e., equal weight) except for the KL term in $\mathcal{L}_{\text{cvae}}$, where we use a weight of 0.01.

After the tracking and forecasting modules are trained, we train the diversity sampling function (DSF) with Adam and a learning rate of 1e-4 for 20 epochs. We keep the learning rate fixed for 10 epochs and then linearly decay the learning rate to 0. The DSF training loss is defined in Eq. (14) of the main paper. The weight for \mathcal{L}_{dpp} is always 1, while the weight for $\mathcal{L}_{\text{recon}}$ is 10 for the KITTI dataset and 50 for the nuScenes dataset. The ω in Eq. (11) is set to 100 for KITTI and 200 for nuScenes. The R in Eq. (11) is set to the radius where 90 percent of the Gaussian samples lies within, which can be found using the percentage point function of the chi-squared distribution.