2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

FDNet: Feature Decoupling Framework for Trajectory Prediction

Yuhang Li[1] , Changsheng Li[1], Baoyu Fan[2], Rongqing Li[1], Ziyue Zhang[1],
Dongchun Ren[3], Ye Yuan[1] and Guoren Wang[1]

Abstract— **Trajectory prediction plays a significant role in autonomous driving, with current challenges primarily focused on capturing complex interactions in traffic scenes. Previous methods usually directly encode non-interactive and interactive information together, and then decode them for trajectory prediction. However, given the complexity inherent property in the trajectory generation process (e.g., the generation of trajectory points are influenced by the interactions among multiple moving agents, as well as the interactions between agents and the static environment), previous approaches fail to precisely capture separate variations of the trajectory generation process. In this paper, we propose a general and plug-and-play feature decoupling framework for trajectory prediction called FDNet, which can learn the interactive and non-interactive factors in the latent space to capture separate variations of the trajectory generation process. At its core, FDNet is comprised of a Non-interactive Feature Extraction Module to extract non-interactive features, and an Interactive Feature Decoupling Module to decouple interactive features. Extensive experiments conducted on Argoverse and nuScenes demonstrate that FDNet significantly improves the performance of existing methods.**

Index Terms— **Trajectory prediction, Feature decoupling, Information bottleneck, Generative Adversarial Network**

I. INTRODUCTION

Trajectory prediction is essential in autonomous driving, as it forecasts the future behaviors of agents on the road to prevent collisions and enhance safety in dynamic driving scenarios. Given that trajectory prediction tasks are inherently complex, with the goals or intents of traffic agents often remaining unknown and their behavior being influenced by complex interactions in multiagent traffic scenarios, it is both crucial and challenging to model, analyze, and effectively utilize the non-interactive and interactive features of these agents. With the advancement of deep learning, efficiently modeling interactive information within scenes has become a focal point for many trajectory prediction models.

Previous methods [1], [2], [3], [4] apply CNNs to directly model bird's eye view images rasterized from scenes for future trajectory prediction. However, these approaches entail high computational costs and have limited receptive fields. Vector-based methods such as VectorNet [5] and TNT [6] employ a series of vectors to represent the observed agent trajectories and map features, treating each vector as a node in a graph and utilizing graph neural networks to model interactive information. Recent approaches based on Transformer [7], exemplified by HiVT [8], have demonstrated outstanding performance in trajectory prediction. HiVT divides the prediction task into two key phases: local context extraction and global interaction modeling, utilizing Transformer to model temporal information and interactions among agents.

Although the quality of the learnt representations gradually improves, the above-mentioned approaches do not place sufficient emphasis on learning disentangled representations. They encode non-interactive and interactive information into mixed features that contain all the information. The task of trajectory prediction is inherently complex, requiring consideration of the interactions among multiple moving agents, as well as the interactions between these agents and the static environment. Consequently, the aforementioned approaches, which merge various feature subspaces together, lead to a degradation in performance. This is primarily because the interactive features among agents are not adequately highlighted and remain underutilized [9], [10]. Therefore, it becomes necessary to acquire disentangled latent variables and capture separate variations of the trajectory generation process. This could encompass semantic meanings, provide an opportunity to eliminate unwanted variations for a lower sample complexity of motion planning for autonomous driving, and facilitate more controllable generations.

To this end, we propose FDNet, a plug-and-play, general and principled feature decoupling framework to extract and separate the interactive and non-interactive factors in the latent space. In our approach, the ground truth trajectories of agents in non-interactive scenarios are obtained through nonlinear fitting methods, which are inaccurate in certain scenarios. Therefore, to extract more accurate non-interactive features, we design a customized Non-interactive Feature Extraction Module based on Generative Adversarial Network [11] to more effectively learn the distribution of non-interactive features. In addition, to decouple the interactive features while retaining as much information as possible, we develop an Interactive Feature Decoupling Module tailored for the trajectory prediction domain, which is based on the Information Bottleneck network [12]. Specifically, we let interactive features preserve maximum information about mixed features while maintaining minimal information about non-interactive features. After decoupling mixed features into non-interactive and interactive features, we design two dedicated decoders to decode them as non-interactive trajec-

[1]Yuhang Li, Changsheng Li, Rongqing Li, Ziyue Zhang, Ye Yuan and Guoren Wang are with the School of Computer Science, Beijing Institute of Technology, China {yuhangli, lcs, lrq, zhang-ziyue, yuan-ye}@bit.edu.cn, wanggrbit@126.com

[2]Baoyu Fan is with the College of Computer Science, Nankai University, China fanbaoyu@mail.nankai.edu.cn

[3]Dongchun Ren is with the ALLRIDE.AI, China dongchun.ren@gmail.com

Corresponding Author: Changsheng Li.

This work was supported by the NSFC under Grants 62122013, U2001211.

tories and their corresponding offsets caused by interactions, respectively. The final trajectories are obtained by adding the decoded offsets to the decoded trajectories. In this study, we employ Vectornet, TNT, and HiVT as the backbone architectures and validate our FDNet on two datasets. The experimental results show that our method significantly enhances the performance of the backbone models.

Our principal contributions can be summarized as follows:

- We propose FDNet, a feature decoupling network tailored for trajectory prediction. FDNet can decouple mixed features containing all information into non-interactive and interactive features, and capture separate variations of the trajectory generation process.
- We devise a Non-interactive Feature Extraction Module based on Generative Adversarial Network to learn the distribution of non-interactive features, and design an Interactive Feature Decoupling Module based on Information Bottleneck network to extract interactive features. This can effectively decouple mixed features into concise and information-dense non-interactive and interactive features.
- We applied our FDNet to three different models across two widely used trajectory prediction datasets, demonstrating its effectiveness and compatibility in trajectory prediction.

II. RELATED WORK

Trajectory prediction. Early works [13], [14] in trajectory prediction are primarily centered on the historical trajectories of agents, utilizing basic social pooling methods to analyze inter-agent interactions. Subsequently, to better utilize map information, rasterized methods [2], [3], [4], [15], [16] were introduced. These methods render high-definition (HD) maps and agent states as color-coded attributes and encode them with Convolutional Neural Networks. However, due to their limited receptive fields, high costs, and suboptimal results, the focus has increasingly shifted towards vectorized methods. Vectorized methods represent agent dynamics and map elements in vectorized form, then utilize Graph Neural Networks [17], [6], [18], [19], [20] and attention mechanisms [21], [22], [23], [24] to extract interactive information within the scene. Moreover, drawing upon the widespread achievements of Transformer models across various fields [25], [26], [27], many approaches [28], [8], [29], [30], [31], [32] adopt Transformer as their encoders to encode the entire scene information, showing excellent performance. However, the aforementioned methods merge various feature subspaces together, often overlooking the interactive features among agents, leading to diminished performance. Therefore, our research focuses on how to decouple mixed features into interactive and non-interactive features during the decoding phase, to more comprehensively model interactive information.

Feature decoupling. In deep learning, many approaches to feature decoupling tend to leverage representation learning. To capture domain-specific information, [33] introduced a decorrelation regularization based on covariance distance. To eliminate the appearance shift component in domain transfer, [34] implemented supervised adversarial training for learning decoupled representations. To align deep features without the requirement for additional annotations, [35] proposed an Unstructured Feature Decoupling Network, comprising a transformer-based feature decomposing head and a novel cluster-based decoupling constraint. To extract and separate the static and dynamic factors in the latent space, [36] employs a contrastive learning approach to maximize the mutual information between latent factors and input data, while simultaneously minimizing the information between the static and dynamic factors. However, these methods are not readily applicable to the domain of trajectory prediction. Therefore, we presents a novel and efficient feature decoupling framework specifically tailored for trajectory prediction, utilizing generative adversarial networks and the information bottleneck model.

III. METHOD

A. Problem Definition

In this work, we denote $X_P = \{x_{-T'+1}, x_{-T'+2}, ..., x_0\}$ as the historical trajectory of the target agent, where T' represents the observed time steps, and $x_i \in \mathbb{R}^2$ denotes the agent's i-th location. Our goal is to forecast its future states $X_F = \{x_1, x_2, ..., x_T\}$ up to a fixed time step T. Considering that vehicles interact with static HD maps M and other agents during transit, we initially encode the input to obtain features V that contain all the information. Then, the overall probabilistic distribution we want to capture can be described as $p(X_F|V)p(V|M, X_P, X_P^O)$, where X_P^O represents the observed trajectories of these other agents. In this paper, we aim to decouple mixed features V into non-interactive features Y and interactive features Z to more effectively capture separate variations of the trajectory generation process and enhance the utilization of interactive information. Therefore, the distribution of our method can be decomposed as $p(X_F|Y, Z)p(Y, Z|V)p(V|M, X_P, X_P^O)$.

B. Overall Framework

Figure 1 illustrates the architecture of the model we propose. Initially, inputs M, X_P, and X_P^O are fed into a backbone structure (for example, HiVT), resulting in the generation of latent features V that encompass all information from the historical scene. Concurrently, the historical trajectories of target agents, X_P, are processed through an LSTM network to produce its non-interactive features, $\widehat{Y}$. To align the feature distribution of Y, generated by V, as closely as possible to $\widehat{Y}$, both Y and $\widehat{Y}$ are input into a Non-interactive Feature Extraction Module. Following this, interactive features Z are generated from V via an Interactive Generator, and we employ an Interactive Feature Decoupling Module to refine Z into features solely containing interactive information. By employing Y, we obtain non-interactive trajectories, and with Z, we calculate the trajectory offsets caused by interactions within the scene. Summing these components yields the final predicted trajectories X_F.

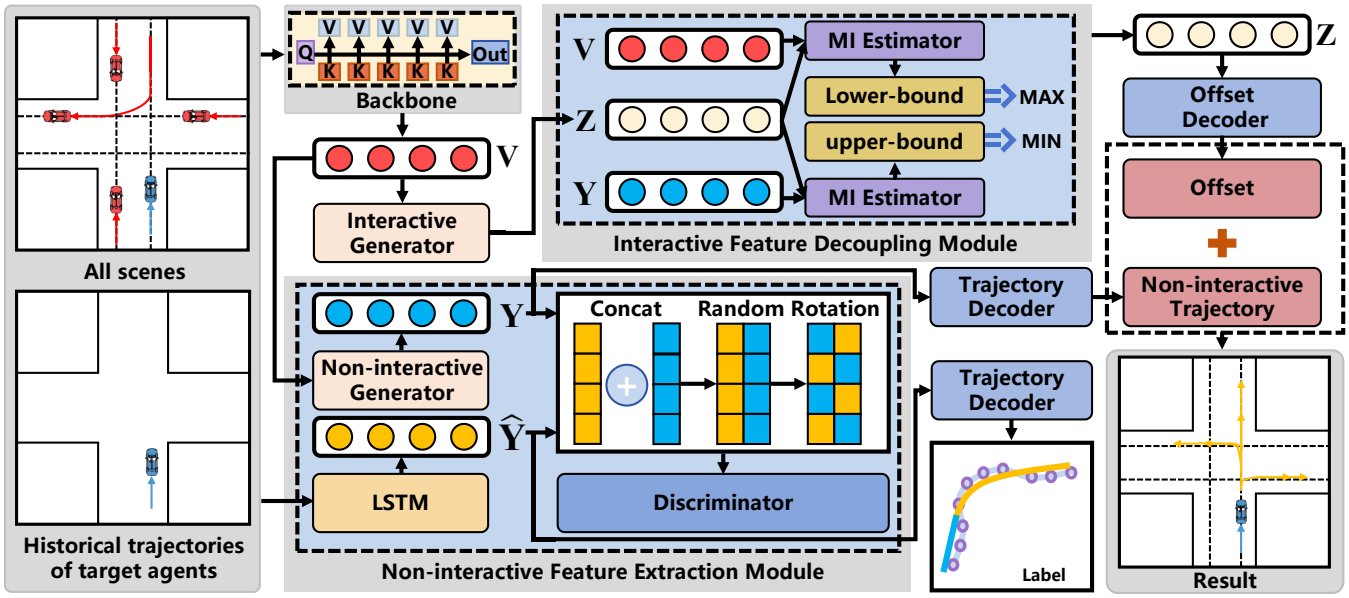

Fig. 1. Overview of FDNet framework. The inputs are the complete information of the scene and the historical trajectory of the target agent. The output are the predicted trajectories of the target agent. Our model serves as a plug-and-play framework for feature decoupling, capable of separating mixed features into non-interactive features and interactive features and capturing separate variations of the trajectory generation process.

C. Extraction of Non-interactive Features

In order to learn the distribution of non-interactive features, we design a specialized Non-interactive Feature Extraction Module based on Generative Adversarial Network. This module is specifically designed to facilitate the learning of more accurate non-interactive feature distributions. First, we input M, X_P, and X_P^O into the backbone Φ, resulting in the generation of latent features denoted as V:

$$V = \Phi(M, X_P, X_P^O; \phi), \tag{1}$$

where the backbone Φ is parameterized by ϕ, and can be an arbitrary trajectory prediction model, e.g., VectorNet [5], TNT [6] and HiVT [8] used in this paper. After inputting V into the Non-interactive Generator G, the feature Y is obtained:

$$Y = G(V; \theta_g), \tag{2}$$

where G is parameterized by θ_g. We aim for the Non-interactive Generator G to capture the distribution of non-interactive features, thereby enabling Y to represent these non-interactive information effectively. Therefore, we also input the historical trajectories of agents into an LSTM encoder to obtain their true non-interactive features $\widehat{Y}$, which are then processed through a Trajectory Decoder D_t to generate the corresponding non-interactive trajectories $\widehat{X}_F$:

$$\widehat{Y} = LSTM(X_P; W_{lstm}), \tag{3}$$

$$\widehat{X}_F = D_t(\widehat{Y}; W_{decoder}), \tag{4}$$

where the parameters of $LSTM$ and D_t are W_{lstm} and $W_{decoder}$, respectively. By employing the method of least squares, we fit the historical trajectories of target agents into a cubic polynomial. This approach enables us to derive pseudo labels for their future non-interactive trajectories. However, while the labels generated using this method accurately represent the future non-interactive trajectories in most scenarios, they are not applicable in certain specific situations. Therefore, to prevent our model from overly focusing on scenarios with erroneous labels and to better capture the global data distribution of non-interactivity, we do not directly replace Y with $\widehat{Y}$. Instead, we design a Non-interactive Feature Extraction Module based on the Generative Adversarial Network to learn Y. The performance of this module will be validated in the subsequent ablation studies.

After obtaining Y and $\widehat{Y}$, we classify them as 'fake' and 'real'. Owing to the variability in historical speeds and travel directions of agents across different non-interactive scenarios, we have developed a novel Generative Adversarial Network tailored for trajectory prediction. This approach is designed to eliminate feature differences caused by varying agents and to enable the model to learn non-interactive characteristics more swiftly and effectively. We concatenate the scene-corresponding Y and $\widehat{Y}$, apply a random flip, and then feed them into the Discriminator, as illustrated in Figure 1. This input can be denoted as $Y_{input} = Flip(Concat(Y, \widehat{Y}))$. A one-dimensional vector e, composed of 0 or 1, is used to indicate whether the features of the scenario have been flipped. The Discriminator D and Non-interactive Generator G play a mini-max game with value function $V(D, G)$:

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{\widehat{y}\sim p_{data}(\widehat{y})}[log\,(D(y,\widehat{y})[1-e])] \\ &+ \mathbb{E}_{y\sim p_y}[log\,(1-D(y,\widehat{y})[e])],\end{aligned} \tag{5}$$

The loss function employed for training the Non-interactive Generator is as follows:

$$\mathcal{L}_g = \mathbb{E}_{y\sim p_y}[log\,(1-D(y,\widehat{y})[e])] \tag{6}$$

D. Decoupling of Interactive Features

After obtaining mixed features V from the backbone and the non-interactive features Y, we develop an Interactive Feature Decoupling Module based on Information Bottleneck model specifically designed for trajectory prediction to decouple the interactive feature Z. Unlike traditional information bottlenecks [12], [37], our model does not rely on Markov chains. We expect Z to preserve maximum information about V while maintaining minimal information about Y, ultimately retaining all of the interactive information. The learning objective of the Interactive Feature Decoupling Module is:

$$R_{IB} = \max I(V, Z) - \beta I(Z, Y), \tag{7}$$

where $I(\cdot, \cdot)$ signifies the mutual information between features. The Lagrange multiplier, denoted as β, facilitates a balance between retaining as much information in Z as possible and minimizing the information from Y.

Given the complexity of directly optimizing Eqn. 7, we draw inspiration from MINE [38], [39] and CLUB [40], [41] to establish two mutual information estimators. These estimators are designed to calculate the lower bound of $I(V, Z)$ and the upper bound of $I(Z, Y)$, respectively.

In calculating the lower bound $I_\psi(V, Z)$ of $I(V, Z)$, we treat mutual information as the Kullback-Leibler divergence between the joint and marginal distributions, transforming it into its dual representation:

$$\begin{aligned} I(V, Z) &\geq I_\psi(V, Z) \\ &= \sup_\psi \mathbb{E}_{p(V,Z)}[T_\psi] - \log\left(\mathbb{E}_{p(V)p(Z)}\left[e^{T_\psi}\right]\right) \end{aligned} \tag{8}$$

where $T_\psi : \mathcal{V} \times \mathcal{Z} \to \mathbb{R}$ is parametrized by a deep neural network with parameters ψ. Then, we calculate the upper bound $I_\omega(Z, Y)$ for $I(Z, Y)$:

$$\begin{aligned} I(Z, Y) \leq & \mathrm{I}_\omega(Z, Y) \\ = & \mathbb{E}_{p(Z,Y)}[\log q_\omega(y \mid z)] \\ & - \mathbb{E}_{p(Z)}\mathbb{E}_{p(Y)}[\log q_\omega(y \mid z)] \end{aligned} \tag{9}$$

where $q_\omega(y \mid z)$ is a neural network used to approximate $p(y \mid z)$. Therefore, we could derive an lower bound J_{IB} of R_{IB}:

$$R_{IB} \geq J_{IB} = I_\psi(V, Z) - \beta I_\omega(Z, Y) \tag{10}$$

We can achieve the objective of feature decoupling by maximizing the lower bound of R_{IB}. Subsequently, we utilize the Monte Carlo sampling to approximate the expectations in J_{IB}, leading to the derivation of the following objective function:

$$\begin{aligned} J_{IB} = & \frac{1}{N} \sum_{i \in B_d} \left[T_\psi\left(v_i, z_i\right) - \log\left(\frac{1}{|N|} \sum_{j \in B_d} e^{T_\psi(v_i, z_j)} \right) \right] \\ & + \frac{\beta}{N} \sum_{i \in B_d} \left[\log q_\omega\left(y_i \mid z_i\right) - \frac{1}{|N|} \sum_{j \in B_d} \log q_\omega\left(y_j \mid z_i\right) \right], \end{aligned} \tag{11}$$

where B_d is the d-th batch of agents and N is the batch size. v_i, y_i and z_i represent the mixed features, non-interactive features, and interactive features of the i-th agent, respectively. The loss function, denoted as $\mathcal{L}_{IB}$, is defined as the negative of J_{IB}. Detailed derivation is provided in the appendix.

After obtaining Y and Z, we feed them into their respective decoders to predict the future trajectories X_{non} of agents under non-interactive conditions and the offset X_{offset} caused by interactions in the scene. By summing these two components, we can derive the final result X_F.

$$\begin{aligned} X_{non} &= D_{non}(Y; W_{non}) \\ X_{offset} &= D_{offset}(Z, X_{non}; W_{offset}) \\ X_F &= X_{non} + X_{offset} \end{aligned} \tag{12}$$

E. Training

We adopt the winner-takes-all strategy on both the non-interactive trajectories and the final trajectories of agents, resulting in a regression loss function denoted as $\mathcal{L}_{reg}$. A classification loss $\mathcal{L}_{cls}$ is also adopted to score each trajectory if the backbone has scoring module and a regression loss $\mathcal{L}_{goal}$ for goals is adopted if the backbone is goal-based. Finally, the total loss function is summarized as follows:

$$\mathcal{L} = \mathcal{L}_{reg} + \alpha \mathcal{L}_{cls} + \gamma \mathcal{L}_{goal} + \delta \mathcal{L}_g + \lambda \mathcal{L}_{IB}, \tag{13}$$

where α, γ, δ and λ are trade-off hyper-parameters. Comprehensive analysis of the hyperparameters used in the loss function are provided in the appendix.

F. The whole algorithm

We provide the pseudo-code of our training procedure, as shown in Algorithm 1. During the training process, each iteration encompasses four gradient updates, including the update of the Discriminator gradient, individual gradient updates for two mutual information estimators, and an update based on the total loss $\mathcal{L}$.

IV. EXPERIMENTS

A. Experiment Settings

Dataset. We evaluated the performance of our method on two widely used datasets: Argoverse [42] and nuScenes [43]. The Argoverse dataset comprises 323,557 real-world driving scenarios, divided into 205,942 training, 39,472 validation, and 78,143 test samples. Both the training and validation sets have an observation duration of 5 seconds at a sampling rate of 10Hz, incorporating 2 seconds of past observation and 3 seconds of future prediction. The test set contains only 2 seconds of observation data. The nuScenes dataset consists of 32,186 training, 8,560 validation, and 9,041 test scenarios. Each scenario spans 8 seconds at a sampling rate of 2Hz, with the initial 2 seconds serving as the observation trajectory and the subsequent 6 seconds as the predictive trajectory.

Baselines and Evaluation Metrics. Given that our model, FDNet, is designed as a plug-and-play system, we employ HiVT [8], VectorNet [5], and TNT [6] as our backbone networks for comparative analysis in our experiments. Consistent with previous studies, we utilize three evaluation metrics: minADE@K, minFDE@K, and minMR@K, where K denotes the number of generated trajectories. In our experimental setup, we set K to 1 and 6.

TABLE I

MINADE@K, MINFDE@K, AND MR@K OF DIFFERENT METHODS ON ARGOVERSE AND NUSCENES DATASET.

Dataset	Methods	K=1			K=6		
		minADE	minFDE	MR	minADE	minFDE	MR
Argoverse	VectorNet [5]	1.5230	3.2907	0.5719	1.0610	1.9005	0.3272
	FDNet+VectorNet	**1.4733**	**3.1804**	**0.5503**	**1.0436**	**1.8651**	**0.3084**
	TNT [6]	1.8514	4.2313	0.6383	0.9465	1.7346	0.2132
	FDNet+TNT	**1.7460**	**3.9471**	**0.6120**	**0.9219**	**1.6757**	**0.2011**
	HiVT [8]	1.7031	3.7744	0.6414	0.6611	0.9691	0.0920
	FDNet+HiVT	**1.2933**	**2.8376**	**0.4812**	**0.6590**	**0.9643**	**0.0912**
nuScenes	VectorNet [5]	3.5811	7.9145	0.8861	2.1284	4.2457	0.7565
	FDNet+VectorNet	**3.3619**	**7.5800**	**0.8612**	**2.0640**	**3.9271**	**0.7252**
	TNT [6]	4.9403	11.8158	0.9208	1.9880	3.9769	0.4877
	FDNet+TNT	**4.4713**	**10.5561**	**0.8901**	**1.9588**	**3.8858**	**0.4787**
	HiVT [8]	4.0945	9.3306	0.8802	1.6666	3.1293	0.4362
	FDNet+HiVT	**3.3923**	**7.8138**	**0.8377**	**1.4152**	**2.5152**	**0.4273**

Algorithm 1 Training Procedure of FDNet

Input: input trajectories X_P and X_P^O, ground-truth trajectory X^F, HD maps m. Four trade-off hyper-parameters: α, γ, δ and λ.

Output: Network parameters: ϕ, ω, θ_g, θ_n, W_{lstm}, $W_{decoder}$, and ψ.

Initialize: Randomly initialize ϕ, ω, θ_g, θ_n, W_{lstm}, $W_{decoder}$, and ψ.

while *not converges* **do**

Compute mixed features $V = \Phi(m, X_P, X_P^O; \phi)$ and latent features $\widehat{Y} = LSTM(X_P; W_{lstm})$;

Compute non-interactive features $Y = G(V; \theta_g)$ and interactive features $Z = D_n(V; \theta_n)$;

Compute predicted trajectory X_F and non-interactive trajectory X_{non} by Y, Z and $\widehat{Y}$;

Compute $\mathcal{L}_{reg}$, $\mathcal{L}_{cls}$ and $\mathcal{L}_{goal}$ through X_F and X_{non};

Concatenate and random flip features Y and $\widehat{Y}$ to get $Y_{input} = Flip(Concat(Y, \widehat{Y}))$;

Compute $\mathcal{L}_g$ to train the Non-interactive Generator;

Compute $\mathcal{L}_{IB}$ using mutual information estimators;

Calculate the total loss $\mathcal{L} = \mathcal{L}_{reg} + \alpha\mathcal{L}_{cls} + \gamma\mathcal{L}_{goal} + \delta\mathcal{L}_g + \lambda\mathcal{L}_{IB}$;

Update model parameters ϕ, θ_g, θ_n, W_{lstm} and $W_{decoder}$ by minimizing $\mathcal{L}$.

Update Discriminator D by optimizing $\max_D V(D, G)$.

Sample $\{(z_i, y_i)\}_{i=1}^N$ from $p_\phi(z, y)$;

Compute log-likelihood $\mathcal{L}(\omega) = \frac{1}{N}\sum_{i=1}^N \log q_\omega(y_i|z_i)$;

Update $q_\omega(y|z)$ by maximizing $\mathcal{L}(\omega)$.

Draw N minibatch samples from the joint distribution: $(v^{(1)}, z^{(1)}), \ldots, (v^{(N)}, z^{(N)}) \sim p_\phi(v, z)$;

Draw N samples from the Z marginal distribution: $\bar{z}^{(1)}, \ldots, \bar{z}^{(N)} \sim p_\phi(z)$;

Evaluate the lower-bound: $\hat{G}(\psi) \leftarrow \hat{\nabla}_\psi \mathcal{V}(\psi)$;

Update the statistics network parameters:$\psi \leftarrow \psi + \hat{G}(\psi)$.

end

TABLE II

ABLATION STUDY ON THE ARGOVERSE VALIDATION SET.

IFDM	NFEM	minADE	minFDE	minMR
		1.7031	3.7744	0.6414
✓		1.3209	2.9124	0.5029
✓	✓	1.2933	2.8376	0.4812

Implementation Details. We utilize the encoders from backbones to capture non-interactive and interactive information and extract mixed features containing all the information from the scenes. Unlike HiVT and TNT, Vectornet does not incorporate a scoring module to evaluate each trajectory. Therefore, in our experiments with Vectornet, we independently trained two models, one for each condition: K=1 and K=6. The feature dimension for HiVT is set to 128, while the other two backbones have a feature dimension of 64. Our model is trained on 8 3090Ti GPUs with a batch size of 64, using the AdamW optimizer for 128 epochs. The initial learning rate, weight decay, and dropout rate are set to 3×10^{-4}, 1×10^{-4}, and 0.1, respectively.

B. Quantitative Results

We evaluate the performance of FDNet on Argoverse and nuScenes using three different backbone models, as listed in Table I. Given that Vectornet lacks the classification function, we additionally trained a separate model specifically for K=1. So Vectornet demonstrates a slightly improved performance at K=1. The results indicate a significant improvement in predictive accuracy when plugging our approach into these models, underscoring the effectiveness and compatibility of our method.

C. Ablation Study

We conducted ablation studies on the Interactive Feature Decoupling Module (IFDM) and the Non-interactive Feature Extraction Module (NFEM) within our model, as illustrated in Table 2. Due to space limitation, we utilize HiVT as the

backbone and evaluate the model's performance with K set to 1. The results indicate that adding the Interactive Feature Decoupling Module significantly enhances model performance, validating its effectiveness for prediction. Moreover, employing the the Non-interactive Feature Extraction Module to learn the distribution of non-interactive features further improves module performance.

D. Qualitative Results

In Figure 2, we visualize the predicted trajectories in four distinct scenarios. It demonstrates that the non-interactive features decoupled by FDNet contain information about the future motion intentions of the agents, enabling the generation of initial predictions with multimodality. The decoupling and decoding of interactive features significantly enhance the model's ability to learn interactive information within the scene. Compared to other methods, FDNet yields more logical and accurate predictions of future trajectories.

V. CONCLUSION

In this paper, we proposed a trajectory feature decoupling framework for trajectory prediction. The proposed method can learn the interactive and non-interactive factors in the latent space to learn disentangled representations, which can capture separate variations of the trajectory generation process. Extensive experimental results demonstrated that this method significantly enhanced the performance of various trajectory prediction models.

VI. APPENDIX

A. Formulaic derivation of IB

To establish the lower bound $I_\psi(V,Z)$ for $I(V,Z)$, we first demonstrate the following formula based on [38]:

$$D_{KL}(\mathbb{P}\,||\,\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log\left(\mathbb{E}_{\mathbb{Q}}[e^T]\right) \tag{14}$$

For a given function T, we construct a Gibbs distribution with probability density function $\mathbb{G}$ as follows:

$$d\mathbb{G} = \frac{1}{Z} e^T d\mathbb{Q}, \tag{15}$$

where $Z = \mathbb{E}_{\mathbb{Q}}[e^T]$. Therefore, we have:

$$\begin{aligned} \mathbb{E}_{\mathbb{P}}[T] - \log\left(\mathbb{E}_{\mathbb{Q}}[e^T]\right) &= \mathbb{E}_{\mathbb{P}}[T] - \log Z \\ &= \mathbb{E}_{\mathbb{P}}\left[\log \frac{d\mathbb{G}}{d\mathbb{Q}}\right] \end{aligned} \tag{16}$$

Let Δ represent the difference between $D_{KL}(\mathbb{P}\,||\,\mathbb{Q})$ and $\mathbb{E}_{\mathbb{P}}[T] - \log\left(\mathbb{E}_{\mathbb{Q}}[e^T]\right)$. This leads to the following derivation:

$$\begin{aligned} \Delta &= D_{KL}(\mathbb{P}\,||\,\mathbb{Q}) - \left(\mathbb{E}_{\mathbb{P}}[T] - \log\left(\mathbb{E}_{\mathbb{Q}}[e^T]\right)\right) \\ &= \mathbb{E}_P\left[\log \frac{d\mathbb{P}}{d\mathbb{Q}} - \log \frac{d\mathbb{G}}{d\mathbb{Q}}\right] \\ &= \mathbb{E}_P \log \frac{d\mathbb{P}}{d\mathbb{G}} \\ &= D_{KL}(\mathbb{P}\,|||\,\mathbb{G}) \end{aligned} \tag{17}$$

Since KL divergence is non-negative, the following formula can be derived:

$$D_{KL}(\mathbb{P}\,||\,\mathbb{Q}) \geqslant \mathbb{E}_{\mathbb{P}}[T] - \log\left(\mathbb{E}_{\mathbb{Q}}[e^T]\right) \tag{18}$$

This lower bound becomes tight when $\mathbb{G} = \mathbb{P}$. Thus, for the optimal function T^*, its form is:

$$T^* = \log \frac{d\mathbb{P}}{d\mathbb{Q}} + C \tag{19}$$

Next, we solve for $I(V,Z)$, which is known to be:

$$\begin{aligned} I(V,Z) &= KL(p(V,Z), p(V)p(Z)) \\ &= \sup_{T:\Omega\to R} \mathbb{E}_{p(V,Z)}[T] - \log\left(\mathbb{E}_{p(V)p(Z)}\left[e^T\right]\right), \end{aligned} \tag{20}$$

where $\Omega = \mathcal{V} \times \mathcal{Z}$ is the input space. Let $\mathcal{F}$ denote the set of functions $T_\psi : \mathcal{V} \times \mathcal{Z} \to \mathbb{R}$ parameterized by a neural network ψ. Defining the lower bound of $I(V,Z)$ as $I_\psi(V,Z)$, we have:

$$\begin{aligned} I(V,Z) \geq\ I_\psi(V,Z) =& \sup_\psi \mathbb{E}_{p(V,Z)}[T_\psi] \\ &- \log\left(\mathbb{E}_{p(V)p(Z)}\left[e^{T_\psi}\right]\right) \end{aligned} \tag{21}$$

To estimate the upper bound of $I(Z,Y)$, we initially set the upper bound based on [40] as follows:

$$\begin{aligned} \mathrm{I}_u(Z;Y) =& \mathbb{E}_{p(Z,Y)}[\log p(y \mid z)] \\ &- \mathbb{E}_{p(Z)}\mathbb{E}_{p(Y)}[\log p(y \mid z)] \end{aligned} \tag{22}$$

By setting $\Delta = I_u(Z;Y) - I(Z;Y)$, it follows that:

$$\begin{aligned} \Delta &= \mathbb{E}_{p(Z,Y)}[\log p(y \mid z)] - \mathbb{E}_{p(Z)}\mathbb{E}_{p(Y)}[\log p(y \mid z)] \\ &\quad - \mathbb{E}_{p(Z,Y)}[\log p(y \mid z) - \log p(y)] \\ &= \mathbb{E}_{p(Z,Y)}[\log p(y)] - \mathbb{E}_{p(Z)}\mathbb{E}_{p(Y)}[\log p(y \mid z)] \\ &= \mathbb{E}_{p(Y)}\left[\log p(y) - \mathbb{E}_{p(Z)}[\log p(y \mid z)]\right] \\ &= \mathbb{E}_{p(Y)}\left[\log(\mathbb{E}_{p(Z)}\left[p(y \mid z)\right]) - \mathbb{E}_{p(Z)}[\log p(y \mid z)]\right] \end{aligned} \tag{23}$$

Given that the logarithmic function $\log(\cdot)$ is concave, the Jensen's inequality implies $\Delta > 0$. Therefore, $I_u(Z;Y)$ serves as an upper bound for $I(Z,Y)$. However, since $p(y|z)$ is unknown, we introduce a variational approximation distribution $q_\theta(y|z)$, modifying the upper bound to:

$$\begin{aligned} \mathrm{I}_\theta(Z;Y) =& \mathbb{E}_{p(Z,Y)}[\log q_\theta(y \mid z)] \\ &- \mathbb{E}_{p(Z)}\mathbb{E}_{p(Y)}[\log q_\theta(y \mid z)] \end{aligned} \tag{24}$$

B. PARAMETER SENSITIVITY

The loss function of FDNet includes five hyperparameters: β, α, γ, δ and λ. The settings for α and γ follow the guidelines provided in the original papers of backbones, where α is set to 1.0 for HiVT [8] and 0.1 for TNT [6], while γ is consistently set to 0.1. Figures 3, 4, and 5 respectively present a sensitivity analysis for β, δ and λ. Figure 3 illustrates the impact of the parameter β on model performance, indicating that the model is relatively insensitive to variations in β. Consequently, we set β to 0.3. Figures 4 and 5 demonstrate that as the hyperparameters increase, the model performance initially declines and then rises. Based on these observations, we set both λ and δ at 0.005.

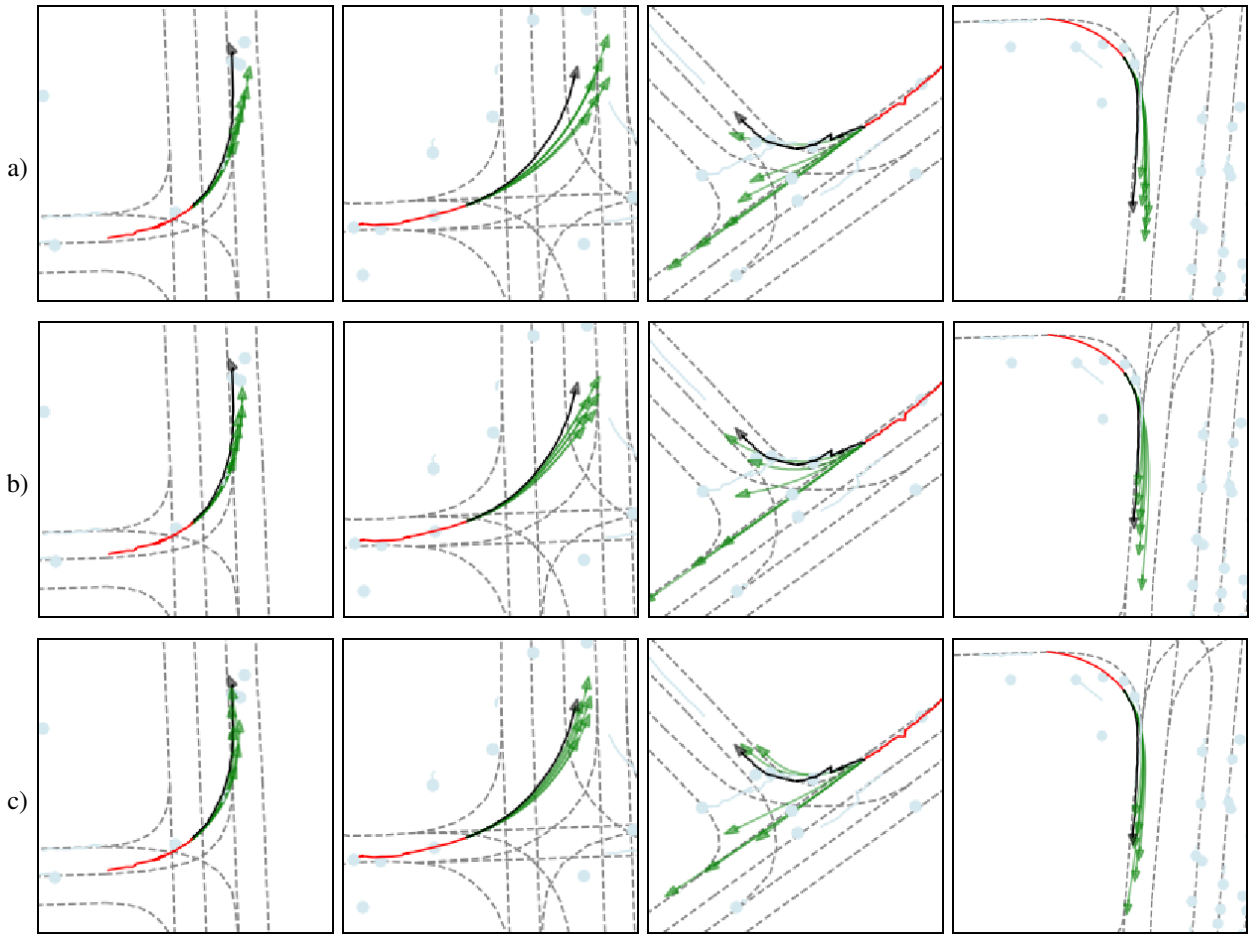

Fig. 2. Visualization of (a) non-interactive trajectories generated by FDNet, predicted trajectories generated by (b) HiVT and (c) FDNet on Argoverse. The past, ground-truth, and predicted trajectories of target agents are depicted in red, black, and green, respectively. In the scene, other agents and the lane centerlines are shown in blue and gray, respectively.

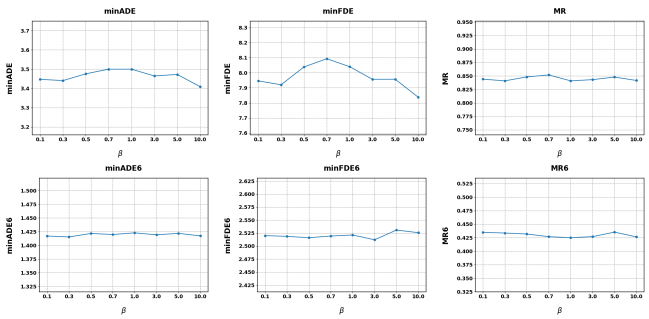

Fig. 3. Parameter sensitivity study about β.

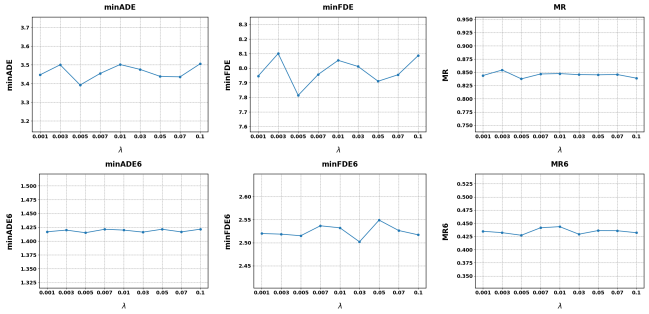

Fig. 4. Parameter sensitivity study about λ.

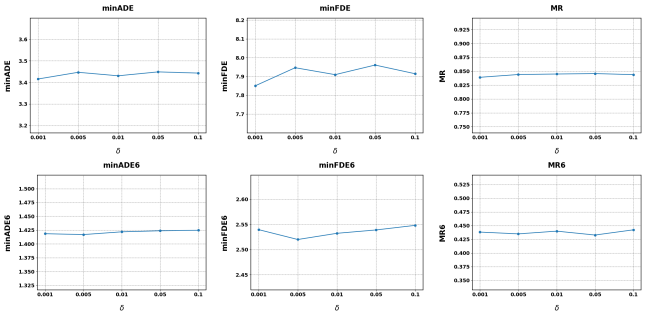

Fig. 5. Parameter sensitivity study about δ.

REFERENCES

[1] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *Conference on Robot Learning*. PMLR, 2018, pp. 947–956.

[2] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv preprint arXiv:1910.05449*, 2019.

[3] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2090–2096.

[4] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8454–8462.

[5] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[6] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, *et al.*, "Tnt: Target-driven trajectory prediction," in *Conference on Robot Learning*. PMLR, 2021, pp. 895–904.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[8] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "Hivt: Hierarchical vector transformer for multi-agent motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8823–8833.

[9] H. Zhang, M. Wang, Y. Liu, and Y. Yuan, "Fdn: Feature decoupling network for head pose estimation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 789–12 796.

[10] J. Sun, Y. Li, L. Chai, H.-S. Fang, Y.-L. Li, and C. Lu, "Human trajectory prediction with momentary observation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6467–6476.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[12] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

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

[14] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[15] S. H. Park, G. Lee, J. Seo, M. Bhat, M. Kang, J. Francis, A. Jadhav, P. P. Liang, and L.-P. Morency, "Diverse and admissible trajectory forecasting through multimodal context understanding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 2020, pp. 282–298.

[16] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo, "Mantra: Memory augmented networks for multiple trajectory prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7143–7152.

[17] S. Kumar, Y. Gu, J. Hoang, G. C. Haynes, and M. Marchetti-Bowick, "Interaction-based trajectory prediction over a hybrid traffic graph," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5530–5535.

[18] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.

[19] A. Cui, S. Casas, K. Wong, S. Suo, and R. Urtasun, "Gorela: Go relative for viewpoint-invariant motion forecasting," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7801–7807.

[20] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. V. Gool, "Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[21] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.

[22] X. Jia, P. Wu, L. Chen, Y. Liu, H. Li, and J. Yan, "Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding," *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[23] X. Wang, T. Su, F. Da, and X. Yang, "Prophnet: Efficient agent-centric motion forecasting with anchor-informed proposals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 995–22 003.

[24] R. Li, C. Li, D. Ren, G. Chen, Y. Yuan, and G. Wang, "Bcdiff: Bidirectional consistent diffusion for instantaneous trajectory prediction," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[25] B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[27] K. Feng, C. Li, D. Ren, Y. Yuan, and G. Wang, "On the road to portability: Compressing end-to-end motion planner for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 099–15 108.

[28] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, *et al.*, "Scene transformer: A unified architecture for predicting multiple agent trajectories," *arXiv preprint arXiv:2106.08417*, 2021.

[29] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6531–6543, 2022.

[30] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2980–2987.

[31] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-centric trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 863–17 873.

[32] R. Li, C. Li, Y. Li, H. Li, Y. Chen, D. Ren, Y. Yuan, and G. Wang, "Itpnet: Towards instantaneous trajectory prediction for autonomous driving," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

[33] S. Liu, L. Zhang, X. Yang, H. Su, and J. Zhu, "Unsupervised part segmentation through disentangling appearance and shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8355–8364.

[34] Z. Song, O. Koyejo, and J. Zhang, "Toward a controllable disentanglement network," *IEEE Transactions on Cybernetics*, vol. 52, no. 4, pp. 2491–2504, 2020.

[35] W. Qian, H. Luo, S. Peng, F. Wang, C. Chen, and H. Li, "Unstructured feature decoupling for vehicle re-identification," in *European Conference on Computer Vision*. Springer, 2022, pp. 336–353.

[36] J. Bai, W. Wang, and C. P. Gomes, "Contrastively disentangled sequential variational autoencoder," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 105–10 118, 2021.

[37] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[38] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International conference on machine learning*. PMLR, 2018, pp. 531–540.

[39] S. Wang, K. Feng, C. Li, Y. Yuan, and G. Wang, "Learning to generate parameters of convnets for unseen image data," *arXiv preprint arXiv:2310.11862*, 2023.

[40] P. Cheng, W. Hao, S. Dai, J. Liu, Z. Gan, and L. Carin, "Club: A contrastive log-ratio upper bound of mutual information," in *International conference on machine learning*. PMLR, 2020, pp. 1779–1788.

[41] K. Feng, C. Li, X. Zhang, and J. ZHOU, "Towards open temporal graph neural networks," in *International Conference on Learning Representations*.

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

[43] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.