Explanatory Generative Trajectory Prediction via Weak Preference Supervision

Yongxi Cao

Master of Science Thesis

Delft University of Technology

Cognitive Robotics (CoR)

Explanatory Generative Trajectory Prediction via Weak Preference Supervision

Master of Science Thesis

For the degree of Master of Science in Robotics at Delft University of Technology

by

Yongxi Cao

August 23, 2024

Supervisor:	Dr. Arkady Zgonnikov
	Julian Schumann
Reader:	Dr. Luca Laurenti
	Dr. Yongqi Dong

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Research reported in this work was partially or completely facilitated by computational resources and support of the the Delft AI Cluster (DAIC) at TU Delft (URL:https://doc.daic.tudelft.nl/), but remains the sole responsibility of the authors, not the DAIC team.

Abstract

For trajectory prediction within autonomous vehicle planning and control, conditional variational autoencoders (CVAEs) have shown promise in accurate and diverse modeling of agent behaviors. Besides accuracy, explainability is also crucial for the safety and acceptance of learning-based autonomous systems, especially in autonomous driving. However, the latent distributions learned by CVAE models are often implicit and thus possess low explainability. To address this, we propose a semi-supervised generative modeling framework, ***PrefCVAE***, which utilizes partially and weakly labelled preference pairs to imbue the CVAE's latent representation with semantic meaning. This approach enables the system to estimate measurable attributes of the agents, and to generate manipulable predictions under the CVAE framework. Results show that incorporating our preference loss allows a CVAE-based model to make conditional predictions using the semantic factor of prediction average velocity. Our augmented framework also does not significantly degrade the baseline accuracy of prediction. Additionally, we show that the latent values learned with PrefCVAE better represent the semantic information contained in the data. Finally, we discuss the potential of this loss design to extend to other machine learning applications beyond trajectory prediction, as well as essential tricks for adaptation of human labeling. We hope that our empirical study offers the broader representation learning community a fresh perspective on inductive bias for disentangled and explainable latent representations in deep generative models. Specifically, we demonstrate that preference pair supervision, a simple and cost-effective approach, can effectively aid in learning semantic meanings for sampling-based generative models like the CVAE.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Trajectory prediction involves forecasting the behaviors of nearby road participants based on their recent motions, a critical component of ensuring safe planning in autonomous driving systems. This task must account for complex and multimodal traffic interactions [1], [2]. Deep generative models, including variational autoencoders (VAEs) [3], [4], generative adversarial networks (GANs) [5], [6], and diffusion models [7], [8], are commonly used due to their ability to generate accurate and diverse prediction in complex scenarios. Among these, conditional VAEs (CVAEs) stand out in modeling the causal relationship between the future trajectory, history observation, and the latent generative factors underlying a dataset [9].

Despite their inherent ability to model causality, CVAEs often lack explainability because they learn implicit latent representations. Most research treats trajectory prediction as a standard regression task, focusing on reconstructing and predicting trajectories (Figure 1-1). Typically, the goal is to predict the most likely trajectory based on behavioral patterns in the dataset, represented by an implicit latent code $\mathbf{z}_i$. However, even perfect test set prediction accuracy is not the ultimate objective of prediction research. Instead, prediction ultimately serves the purpose of safe planning [10]. For instance, if a vehicle follows an unseen pattern not present in the dataset, a non-explanatory model trained on limited data may fail to predict accurately, potentially leading to accidents. This challenge underscores the need for explanatory generative trajectory prediction, which aims to learn the key semantic generative factors underlying the data and base predictions on these factors, rather than relying entirely on implicit ones. This semantic causal inference capability is valuable in several ways. For example, (i) In highly dynamic and uncertain environments, a safe planner should consider multiple semantically plausible patterns instead of merely fitting the most likely case indicated by the data, allowing it to make the safest ego plan; (ii) When perception and/or scene encoding modules are faulty, the prediction module should rely on semantic conditions provided by the user or hard-coded into the planner.

Several works have explored causal and semantically meaningful motion representations using generative models. One notable example is DiversityGAN [5], which learns a low-dimensional semantic latent space with moderate human annotation, where each dimension can represent distinct driving maneuvers, such as merging or turning. Similarly, Interpretable Self-aware

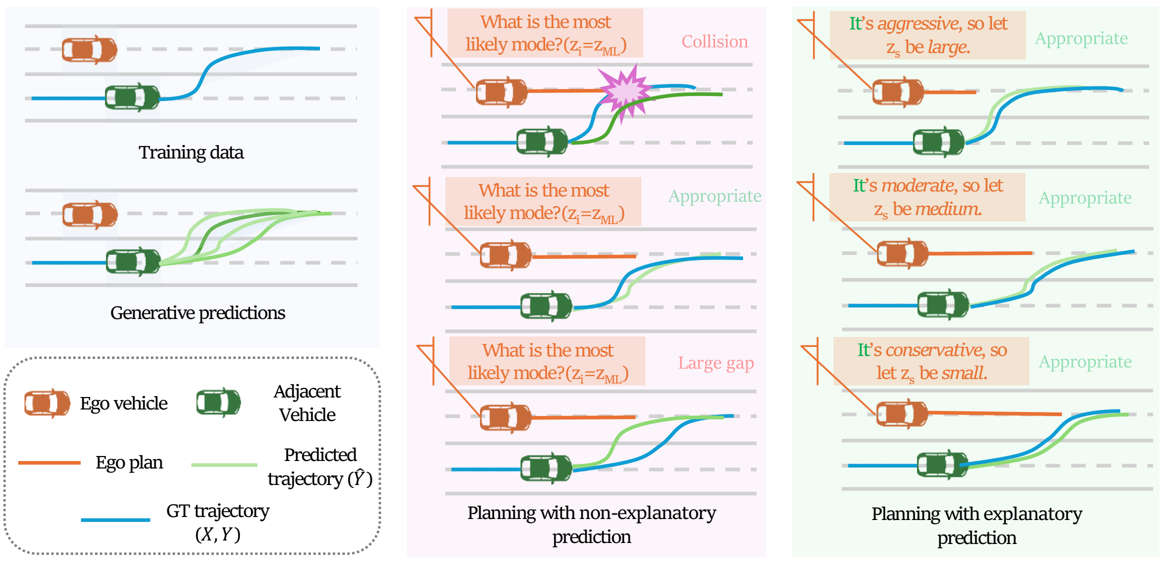

Training data
Generative predictions
Ego vehicle
Adjacent Vehicle
Ego plan
Predicted trajectory ($\hat{Y}$)
GT trajectory (X, Y)
What is the most likely mode?(z_i=z_{ML})
Collision
What is the most likely mode?(z_i=z_{ML})
Appropriate
What is the most likely mode?(z_i=z_{ML})
Large gap
Planning with non-explanatory prediction
It's *aggressive*, so let z_s be *large*.
Appropriate
It's *moderate*, so let z_s be *medium*.
Appropriate
It's *conservative*, so let z_s be *small*.
Appropriate
Planning with explanatory prediction

Figure 1-1: Motivation of explanatory prediction: Most-likely (ML) prediction is not always the most accurate one. To account for multimodal futures, explanatory prediction should reason about the interaction semantically and make corresponding predictions.

Prediction (ISAP) [11] models uncertainty over semantically interpretable latent concepts, including past behavior, map information, and social context. Another approach, Causal Motion Representation (CMR) [12], divides the latent space into three categories: domain-invariant, domain-specific, and non-causal spurious features. Specifically, a contrastive loss is applied to learn the domain-specific style confounder, enabling discrimination based on semantic metrics. However, we identify several limitations in these previous works. While sampling the low-rank latent space of DiversityGAN produces semantically diverse trajectories, the model lacks a structural understanding of the latent space. As a result, predictions cannot be manipulated by directly assigning specific latent values. Furthermore, sampling followed by rejection sampling with additional modules is not efficient enough for practical safe planning. For ISAP and CMR, their unsupervised frameworks also make it challenging to learn consistent semantic representations that align with predefined semantics.

Thus, it remains an open challenge to develop an explanatory generative trajectory prediction model that can *both* explicitly manipulate predictions and interpret trajectories based on predefined semantic factors—this is the focus of our work. To address this, we introduce a weakly supervised augmentation to existing CVAE frameworks called Preference CVAE (PrefCVAE). PrefCVAE leverages partially labeled external data to efficiently learn a semantically meaningful latent space. The core idea is to impose preferences on diverse predictions conditioned on sampled latent values and use ranking to regularize the CVAE. In our view, explanatory prediction has two main desiderata (Figure 1-2): (i) The explanatory representation should enable the understanding of a trajectory concerning specific semantic factors using the CVAE's approximate posterior encoder; (ii) It should generate manipulable and monotonic trajectories with the decoder, conditioned on these known continuous semantic factors by assigning latent variables.

In summary, we argue that incorporating semantic meanings into the latent generative factors of CVAEs can enhance the explainability of trajectory prediction and ultimately contribute

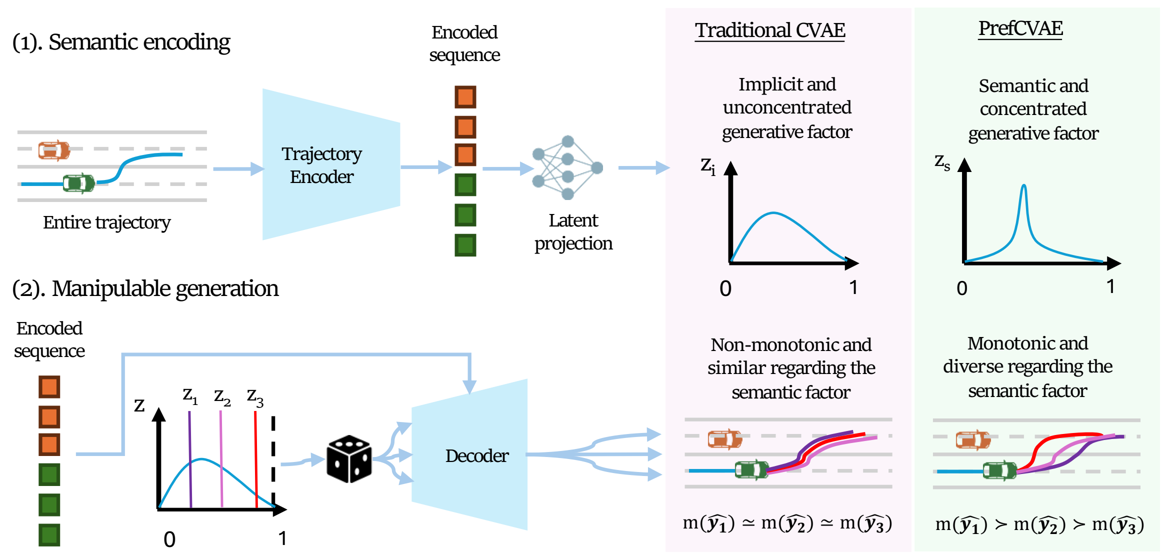

(1). Semantic encoding
Entire trajectory
Trajectory Encoder
Encoded sequence
Latent projection
(2). Manipulable generation
Encoded sequence
z
z1 z2 z3
0
1
Decoder
Traditional CVAE
Implicit and unconcentrated generative factor
zi
0
1
Non-monotonic and similar regarding the semantic factor
m(ŷ1) ≃ m(ŷ2) ≃ m(ŷ3)
PrefCVAE
Semantic and concentrated generative factor
zs
0
1
Monotonic and diverse regarding the semantic factor
m(ŷ1) ≻ m(ŷ2) ≻ m(ŷ3)

Figure 1-2: Our desiderata of explanatory prediction: (i) Use the latent distribution to give an estimation regarding the semantic factor; (ii) Generate manipulable predictions that pertain to latent values. ($m(\cdot)$ represents a generic quantitative semantic metric, and $\succ$, $\simeq$ denote magnitude relationships of "greater than" and "approximately equal", respectively)

to safer and more trustworthy planning [13]. With our PrefCVAE framework, we augment the well-established CVAE-based trajectory prediction model, AgentFormer [4]. The AgentFormer model trained with our PrefCVAE loss demonstrates the ability to predict using a monotonic metric when assigning traversed semantic latent values. Additionally, the approximate posterior encoder more accurately encodes a given trajectory to its ground truth latent value with higher likelihood. These successes indicate that we have achieved a preliminary explanatory generative prediction model.

Chapter 2

Related Works

Explanatory trajectory prediction. In the autonomous driving community, explanatory trajectory prediction is explored from various perspectives. One line of research focuses on integrating prediction directly into planning by jointly optimizing predicted trajectories and ego-vehicle plans, rather than employing a cascading approach [14], [15]. For instance, Task-informed Motion Prediction (TIP) [16] addresses the limitations of task-agnostic predictions by incorporating ego-vehicle plans through the optimization of a task utility function in addition to the traditional trajectory fitting loss. Another approach involves learning transferable motion styles. Inspired by parameter-efficient fine-tuning methods in vision and language processing [17], [18], [19] and [20] employ low-rank modules to enhance the generalizability of prediction models. Additionally, recent advancements in foundation models, such as large language models and vision-language models, have led to the exploration of natural language encoding for trajectory prediction [21]–[24].

Disentanglement and manipulation of latent representations. Our work aligns with ongoing efforts in machine learning to disentangle and manipulate latent spaces. We define disentanglement as the process of learning semantically factorized latent representations, which is a key objective in generative representation learning [25], [26]. In the context of VAEs, the focus of disentanglement has shifted from unsupervised learning [27] to semi-supervised approaches [28], [29]. Early works such as FactorVAE [30] and β-TCVAE [31] concentrated on reducing the total correlation of latent dimensions, thereby maximizing the mutual information between latent and data distributions. More recent efforts have explored diverse forms of inductive biases that can be effectively leveraged to learn semantic latent spaces [32]–[34]. In image synthesis, GAN-based methods have also advanced disentanglement through techniques like subspace projection [35], style channel discovery and identification [36], and other unsupervised style transfer methods [37], [38]. Our work contributes to the machine learning community as an empirical study of a novel disentanglement method.

Learning from preference. Our approach is conceptually closest to the idea of learning from preference. Reinforcement Learning with Human Feedback (RLHF) has proven to be an effective method for reward approximation and intention alignment. RLHF optimizes a neural reward function that mimics human preferences based on policy rollouts, as well as the policy itself. Initially proposed to learn implicit rewards that are difficult to define in closed form, RLHF has been successful in applications such as gait generation and Atari games [39], [40]. It has since become a key fine-tuning technique for large language models, aligning them with human preferences [41]–[43]. This fine-tuning process can be seen as specifying preferred domains within a large dataset distribution. Recently, DPO [44] relaxed the requirements of reinforcement learning by introducing a supervised framework that resembles RLHF, offering a more stable and data-efficient approach.

Our work differs from the previous works with similar objectives, such as [34], [44], in that ours is a pure end-to-end CVAE-based approach to explanatory trajectory prediction that requires no additional learnable modules, and we focus on injection of preference within the latent distribution of the CVAE. Also, the weak labels our method require are typically much more efficient to acquire than the latent variables' ground truth themselves in many tasks such as imitation learning and data generation [39], [43].

Chapter 3

Background: Trajectory Prediction with CVAE

3-1 Trajectory Prediction: Problem Formulation

In the context of autonomous driving or mobile robots, the objective of trajectory prediction is to predict future trajectories of multiple agents in a scene given their past trajectories observations and other information about the scenario. Consider a multiple agent case, the minimal unit for data representation is a minibatch. A minibatch contains the past trajectory state of agents within the scene, defined as $\mathbf{X} = \left[\boldsymbol{X}^1, \boldsymbol{X}^2, ..., \boldsymbol{X}^{T_{\text{cur}}}\right]$, where $\boldsymbol{X}^t = \left[\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, ..., \boldsymbol{x}_{N(t)}^t\right]$ represents the states of time-variant number of $N(t)$ agents at timestep $t \in \{1, 2, ..., T_{\text{cur}}\}$. The exact state information $\boldsymbol{x}_n^t$ is a versatile design choice, ranging from position, velocity, and acceleration to additional parameters like heading angles or agent types. Besides agent-wise statistics, contextual information ($\mathbf{C}$) is considered, encompassing semantic details of the road (e.g., locations and labels for lanes, sidewalks, etc.). The goal of trajectory prediction is to infer a prediction $\hat{\mathbf{Y}}$ of a future ground truth $\mathbf{Y} = \left[\boldsymbol{Y}^{T_{\text{cur}+1}}, \boldsymbol{Y}^{T_{\text{cur}+2}}, ..., \boldsymbol{Y}^{T_{\text{end}}}\right]$ given $\mathbf{X}$ and optionally $\mathbf{C}$, where $\boldsymbol{Y}^t = \left[\boldsymbol{y}_1^t, \boldsymbol{y}_2^t, ..., \boldsymbol{y}_{N(t)}^t\right]$ is the future trajectory consisting of positions and optionally other information such as velocity and heading angles. For brevity of notation, we cluster the contextual information $\mathbf{C}$ into $\mathbf{X}$, i.e., for following notations $\mathbf{X}$ is essentially $\{\mathbf{X}, \mathbf{C}\}$. Hence, a dataset with K minibatches used for training trajectory prediction models is of the form $\mathbf{D} = \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^K$. Specifically, for deep generative model-based methods, the trajectory prediction objective is to learn a plausible conditional distribution $\hat{\mathbf{Y}} \sim p_\theta(\mathbf{Y}|\mathbf{X})$ from the dataset, where θ implies that the distribution is parameterized with a neural network.

3-2 CVAE Framework for Trajectory Prediction

Conditional VAE (CVAE) provides a causal inference framework, assuming the existence of an M-dimensional generative latent factor $\mathbf{z}_i = [\mathrm{z}_{i,0}, \mathrm{z}_{i,1}, ..., \mathrm{z}_{i,m-1}]$ for the i-th agent in the minibatch, in which elements are random variables and are assumed to dominate characteristics of the motion. For clarity, we sequentialize the latent factor for a minibatch as a vector, $\mathbf{z} = \left[\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{N(t)}\right]$, instead of a matrix. We also denote the minibatch-wise history observation and future prediction as $\mathbf{x}$ and $\mathbf{y}$ to indicate they are as well treated as random variables in the CVAE framework, and can be sequentialized as vectors instead of tensors. As for the trajectory prediction task, the history observation $\mathbf{x}$ is the explicit conditional label variable, and the future trajectory $\mathbf{y}$ is the target data to be reconstructed or predicted.

In the CVAE framework, three modules are learned simultaneously for the tasks of reconstruction and generation of conditioned data (the *prior encoder* $p_\theta(\mathbf{z}|\mathbf{y})$, the *decoder* $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{z})$), and recognition of conditioned data (*approximate posterior encoder*, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$). The encoder distributions are typically obtained by firstly encoding the history observation to a context sequence, and then using an MLP to project to parameters of probability distributions, and decoder distributions are often GMMs or simply a unimodal Gaussian distribution, in which case max likelihood regresses to an MSE objective. Regarding each trajectory, the basic single-datapoint Evidence Lower Bound (ELBO) loss for CVAE's, which is to be minimized for training the CVAE framework, is defined as the sum of negative log-likelihood (reconstruction loss) of the future trajectory $\mathbf{y}$ and KL-divergence between the prior encoded distribution $p_\theta(\mathbf{z}|\mathbf{x})$ and approximate posterior encoded distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$,

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}, \mathbf{y}; \phi, \theta) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log\ p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{z})\right] + \boldsymbol{D}_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})\right]. \tag{3-1}$$

A more detailed note on notation and derivation of CVAE can be found in Appendix.A.

Chapter 4

PrefCVAE

In this section, we describe our preference-based CVAE, the PrefCVAE, framework, still regarding each minibatch. We assume two types of latent variables, P semantically meaningful and measurable ones $\mathbf{z}_S = \left[\mathbf{z}_{S0}, \mathbf{z}_{S1}, ..., \mathbf{z}_{S(P-1)}\right]$, and $M - P$ domain-invariant ones $\mathbf{z}_I = \left[\mathbf{z}_{I0}, \mathbf{z}_{I1}, ..., \mathbf{z}_{I(M-P+1)}\right]$, where each element $\mathbf{z}_{Si}$ or $\mathbf{z}_{Ii}$ is an N-dimensional vector because there are N agents in the minibatch. Each dimension of $\mathbf{z}_S$ controls one semantic metric that could be quantified and evaluated using the trajectory. On the other hand, factors in $\mathbf{z}_I$ pertain to noises or abstract factors that does not semantically contribute to the causal inference. Note that $\mathbf{z}_I$ may still contain useful but implicit information. The objective of PrefCVAE is to explicitly learn the latent variables factorised as $\mathbf{z} = [\mathbf{z}_S, \mathbf{z}_I]$ with weak preference labels.

4-1 Weakly Labeling Preference

This section describes the definition of the extra random variables our method uses, and how they are obtained at training time. For simplicity of notation, we only consider one semantic latent factor in the following procedures, and assumes all other latent dimensions to be domain-invariant, i.e., $\mathbf{z} = \mathbf{z}_S = \{\mathrm{z}_1, \mathrm{z}_2, ..., \mathrm{z}_N\}$ is the only semantic latent factor within a minibatch of data, consisting of N agents' interaction. However, the framework could be easily extended to more than one semantic dimension.

When we want to label preference for different predictions of this agent within the minibatch, we firstly draw two set of additional latent samples uniformly from the latent domain, namely, $\mathbf{z}^0$ and $\mathbf{z}^1$. That is, $\mathbf{z}^0, \mathbf{z}^1 \sim \boldsymbol{U}(\mathbf{z}_{S,\min}, \mathbf{z}_{S,\max})$, where $\mathbf{z}_{S,\min}$, $\mathbf{z}_{S,\max}$ are specific boundaries that may vary for different semantic factors. Since the position of each term $\mathbf{z}_i^0$ and $\mathbf{z}_i^1$ are symmetrical, we fix all $\mathbf{z}_i^0 < \mathbf{z}_i^1$ while sampling, to make loss implementation easier without any compromise. The $\mathbf{z}_I$ is drawn with the same approach, but we do not need to fix the magnitude relationship between each pair of elements $\mathbf{z}^0_{Ij_i}$ and $\mathbf{z}^1_{Ij_i}$.

Secondly, using the CVAE decoder, we make two predictions $\hat{\mathbf{y}}^0$ and $\hat{\mathbf{y}}^1$ by taking expectation of the predicted distribution given the sampled $\mathbf{z}^0$ and $\mathbf{z}^1$, implicit latent factors $\mathbf{z}_I$, and the history observation $\mathbf{x}$. That is, $\hat{\mathbf{y}}^j = \mathbb{E}\left[p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{z}_S, \mathbf{z}^j)\right]$.

Thirdly, we label the ground truth preference $\hat{P}[\hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1]$. We currently use an oracle program, $m(\mathbf{x}, \hat{\mathbf{y}}^i)$, to calculate the metric of each prediction $\hat{\mathbf{y}}^0$ and $\hat{\mathbf{y}}^1$, and give a preference over them. The preference value is essentially an approximation to one of the two weighted latent samples $\{\frac{\mathrm{z}_i^0}{\mathrm{z}_i^0+\mathrm{z}_i^1}, \frac{\mathrm{z}_i^1}{\mathrm{z}_i^0+\mathrm{z}_i^1}\}$, and it indicates which of the two generated predictions *should* have a larger metric value related to this latent factor.

Assuming the closed-form oracle metric of trajectory $(\mathbf{x}, \mathbf{y}^i)$ to be $\hat{m}(\mathbf{x}, \mathbf{y}^i)$, the agent-wise preference between two predicted trajectories $\hat{\mathbf{y}}^0$ and $\hat{\mathbf{y}}^1$ is given by

$$\hat{P}[\hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1; \mathrm{z}_i^0, \mathrm{z}_i^1, \mathbf{x}] = \frac{1}{\mathrm{z}_i^0 + \mathrm{z}_i^1}[(\mathrm{z}_i^1 - \mathrm{z}_i^0)\ \sigma(\eta(\hat{m}(\mathbf{x}, \hat{\mathbf{y}}^0) - \hat{m}(\mathbf{x}, \hat{\mathbf{y}}^1))) + \mathrm{z}_i^0], \tag{4-1}$$

or denote as $\hat{P}[\hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1]$ for short, where $\sigma(\cdot)$ is the Sigmoid function, and η is a scaling factor controlling sensitivity of the oracle preference to difference between two predictions. This is a soft version of *if-else* clause that rolls out one estimation from $\{\frac{\mathrm{z}_i^0}{\mathrm{z}_i^0+\mathrm{z}_i^1}, \frac{\mathrm{z}_i^1}{\mathrm{z}_i^0+\mathrm{z}_i^1}\}$, whose major purpose is to guarantee differentiability in back-propagation. An important design here is, we use Sigmoid function to approximate the discrete values of $\{\frac{\mathrm{z}_i^0}{\mathrm{z}_i^0+\mathrm{z}_i^1}, \frac{\mathrm{z}_i^1}{\mathrm{z}_i^0+\mathrm{z}_i^1}\}$. Otherwise, the *if-else* would be required to distinguish the order of magnitude between $\hat{m}(\mathbf{x}, \hat{\mathbf{y}}^0)$ and $\hat{m}(\mathbf{x}, \hat{\mathbf{y}}^1)$, which makes an inconsistency in the gradient flow. The hard version of choosing from set would make the preference oracle useless. The smoothness of Sigmoid function allow those comparisons who are not distinctive to be equivocal since the output is close to $\frac{1}{2}$ if the two samples are very close.

While the hats over P and m indicate that they are estimated by the oracle program, it is nonetheless still regarded as the ground truth since the framework assumes that the oracle is absolutely correct and noiseless. In this work we show results with preference pairs labelled with oracle program only, and discuss techniques toward adapting human labels. At inference time we also use the same oracles for evaluation. It is beyond the scope of this work to consider wrong or noisy oracles.

4-2 Preference Loss

To fulfill the objective of semantic latent factors, we take an approach that aligns the order of magnitude of metric values with that of the latent factors. The preference loss is a cross entropy between the sampled latents' distribution and the distribution of scores given by the oracle program. The oracle preference plays a role like a ground truth decoder of the metric, and the original CVAE's decoder is like an encoder here. The preference loss is desired to learn to rank two predictions correctly. With the ground truth preference $\hat{P}[\hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1]$ and the two auxiliary sampled latent factors z_i^0 and z_i^1, the agent-wise preference loss is defined as

$$\mathcal{L}_{\text{pref}}(\mathbf{x}, \mathbf{z}_i^0, \mathbf{z}_i^1, \hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1; \phi, \theta) = -[\hat{P}[\hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1]\log(\mathbf{z}_i^0) + (1 - \hat{P}[\hat{\mathbf{y}}^0, \hat{\mathbf{y}}^1])\log(\mathbf{z}_i^1)]. \tag{4-2}$$

In this formula, when $\mathbf{z}_i^0 < \mathbf{z}_i^1$ is fixed, if $m(\hat{\mathbf{y}}^0)$ is larger than $m(\hat{\mathbf{y}}^1)$, the loss value would be large. While if $m(\hat{\mathbf{y}}^0)$ is smaller, the loss would be small. Hence, the preference loss encourages the predicted trajectory to align with the latent in a user-given way: the prediction with smaller latent value is encouraged to have a smaller metric value, and the converse is punished. Note that by swapping the positions of $\mathbf{z}_i^0$ and $\mathbf{z}_i^1$ in the loss function, or fixing $\mathbf{z}_i^0 > \mathbf{z}_i^1$ instead of $\mathbf{z}_i^0 < \mathbf{z}_i^1$ when sampling, we can reverse the way preference aligns with $\mathbf{z}$ value.

Compared to similar loss function designs in [32], [34], the key superiority of ours is that the ground truth latent factor values $\mathbf{z}_{\text{GT}}^0$ and $\mathbf{z}_{\text{GT}}^1$ pertaining to the two predictions $\hat{\mathbf{y}}^0$ and $\hat{\mathbf{y}}^1$ are not required. Instead, the oracle only informs the loss if the relationship of these predictions are correct given the sampled latent factors. This form of weak label broadens the application range of the algorithm. Also, since we leverage preference, it is not assumed that the distribution boundary of latent factors are determined by the dataset. That is, due to stochastic sampling and generation procedure, we may generate unseen yet sensible trajectories in accordance with the latent semantic factors.

For training, we simply use the preference loss alongside the original CVAE ELBO loss, i.e.,

$$\mathcal{L} = \mathcal{L}_{\text{ELBO}} + \lambda \mathcal{L}_{\text{pref}}, \tag{4-3}$$

where λ is a weighting factor, and the minibatch-wise loss is an aggregation of the agent-wise loss.

Chapter 5

Results

Section 5-1 describes the dataset and base model used in our experiments. Section 5-2 presents three major findings and discusses the impact of reducing the use rate, ν, a specific hyperparameter detailed in that section. We demonstrate the effectiveness of PrefCVAE in explanatory trajectory prediction, considering both manipulable trajectory prediction and concentrated latent factor encoding. In Section 5-3, ablation studies confirm that the observed effects are indeed due to the new loss design. We also examine the impact of two other hyperparameters: the weighting factor λ and the latent dimension nz.

5-1 Experimental Setup

Semantic metrics and latent distribution designs. We conduct experiments focusing on a simple low-level semantic metric: the average velocity of the predicted trajectory. Although the plausible range of velocities is not explicitly defined by the dataset, it is assumed that the average velocity socially acceptable for samples in the dataset is bounded. In other words, while we do not assign specific ground truth latent values for each sample during training, the dataset-wise velocity should have an upper bound (speed limit) and a lower bound (static). Therefore, we adopt a bounded distribution, the Beta distribution, as the latent variable to learn the low-level semantic. In Appendix B-3, we also explore the potential use of a high-level semantic, social value orientation (SVO). A brief description of these semantics' formulations and the corresponding latent distributions' formulas is provided in Appendix B-4.

Dataset and base model. Experiments are conducted on the nuScenes trajectory prediction task. We follow the convention of predicting motion for the subsequent 6 seconds (12 frames) based on observations from the past 2 seconds (4 frames). We adopt a widely recognized Transformer-based, multi-agent CVAE trajectory prediction model, AgentFormer [4], as the baseline method. This model is chosen for its effectiveness in capturing multi-agent spatio-temporal relationships in traffic scenes using a unified transformer architecture, while strictly adhering to the CVAE probabilistic framework.

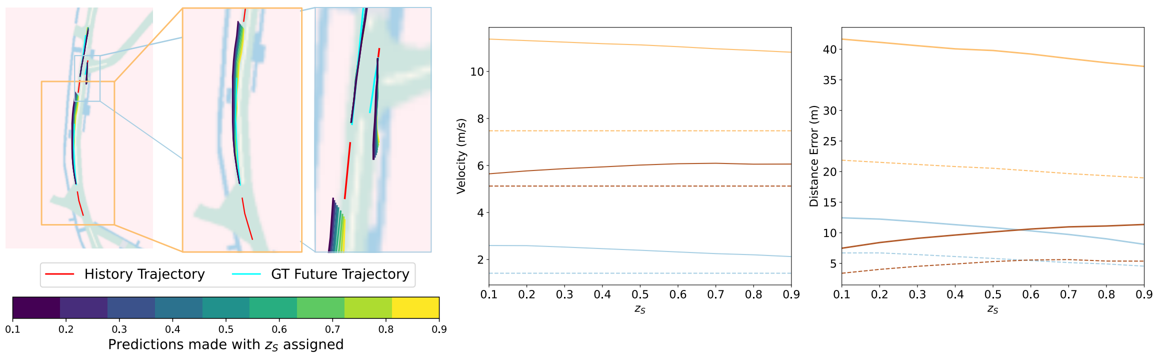
Table 5-1: Accuracy and semantic diversity of base β-AgentFormer and the same model trained with PrefCVAE loss and different use rates. All ADE/FDE's are obtained without assigning latent at test time, and the velocity spans are with latent factor traversal from 0.1 to 0.9 with step size of 0.1.

	Use rate, ν		
	0 (Base)	0.25	1
minADE$_5$ (m)	2.62	2.66	2.64
minFDE$_5$ (m)	5.62	5.76	5.74
Vel. Span (m/s)	3.16	1.87	1.63
(min/max)	3.17	4.38	4.86
Monotonic	No	Yes	Yes

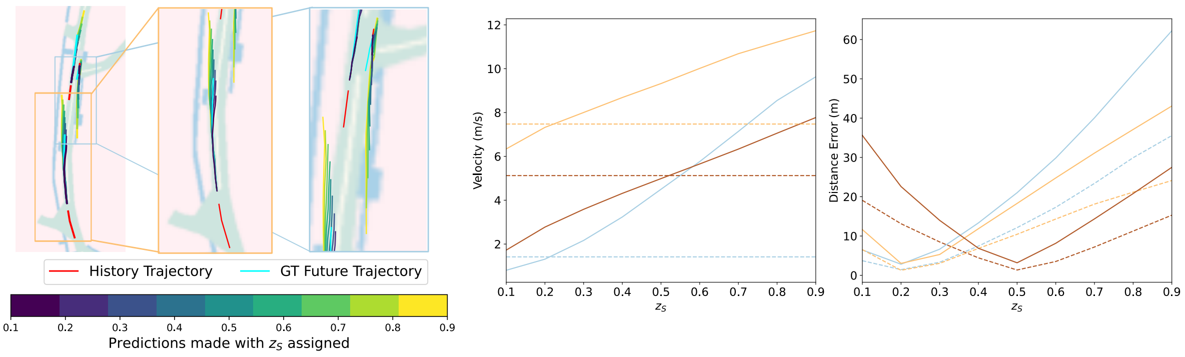
In this work, we use a slightly modified version of AgentFormer. We replace the Gaussian latent distribution with a Beta distribution, where the concentration parameters α and β are clipped to be larger than 1 (as discussed in Appendix B-4). This adjustment enables continuous manipulation of the prediction by traversing within the bounded domain of the random variable definition, $[0, 1]$. Additionally, the latent dimension is reduced from 32 to 2 to minimize randomness and clearly demonstrate the effect of our method. For the remainder of this paper, we refer to this modified version as β-AgentFormer to avoid confusion. A vanilla β-AgentFormer is trained using the ELBO loss and a variety loss defined in the original work (base loss). Lastly, to reduce the impact of the trajectory sampler, we do not apply DLow in the post-training process, as described in the original AgentFormer paper.

All models are trained from scratch using either base loss or PrefCVAE loss for 30 epochs on the entire nuScenes trajectory prediction dataset, with no pretraining applied. The same task-specific auxiliary loss terms are applied to all experiments (see Appendix B-2). The base model used in all results is our modified β-AgentFormer, not the original AgentFormer.

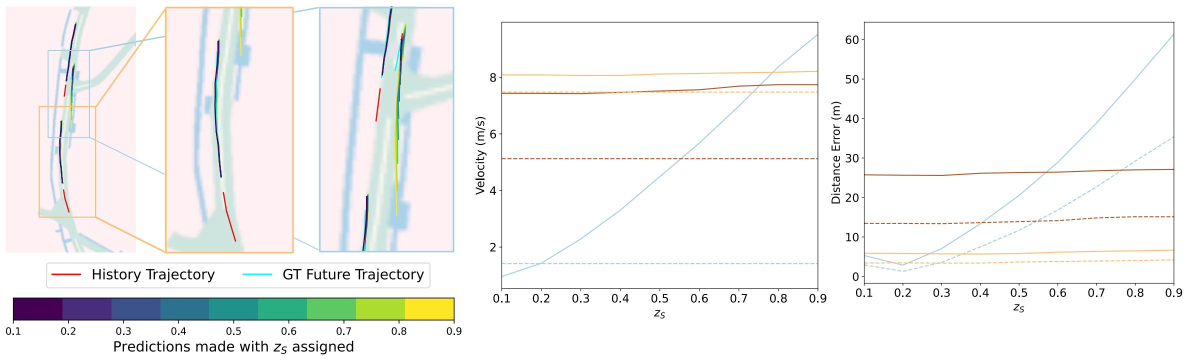
Repeatability. Variational generative models like the CVAE can exhibit stochastic performance due to different neural network initializations [28]. Specifically, for our loss design, certain random seeds may lead to suboptimal performance for β-AgentFormer. To address this, we experiment with multiple random seeds to achieve satisfactory results for each configuration. Our goal is to minimize randomness in the training process and maximize the repeatability of our results. Therefore, all results presented in the tables and figures are obtained as follows: We select three random seeds (specifically, 42, 37, and 43), train the model with each, and record the best outcome based on the relevant metric. If the three results are marginally similar, we default to using the result with seed 42. This approach balances repeatability and performance, demonstrating that our method is not highly sensitive to factors such as neural network initialization and sampling order. Further details on repeatability can be found in Appendix B-1.



(a) Vanilla β-AgentFormer (Test time: traverse all agents)



(b) β-AgentFormer trained with PrefCVAE (Test time: traverse all agents)



(c) β-AgentFormer trained with PrefCVAE (Test time: traverse only one agent)

Figure 5-1: Manipulable predictions, demonstrating usefulness of PrefCVAE framework. For each subfigure, Left: prediction results; Middle: semantic metric w.r.t. z value (horizontal dashed lines are ground truth values); Right: ADE (solid)/FDE (dashed). PrefCVAE can semantically manipulate the prediction: For model trained with PrefCVAE, larger z value always leads to larger average velocity, as learned with the preference loss. The best accuracy occurs around the latent values that pertain to the ground truth velocity (indicated by the dashed horizontal lines).

5-2 Explanatory Trajectory Prediction

PrefCVAE decoder generates manipulable and plausible trajectory predictions. Figure 5-1 illustrates a minibatch from the test set. We evaluate the models by traversing the latent space from 0.1 to 0.9 with a step size of 0.1. In the vanilla AgentFormer model (Figure 5-1(a)), the latent factors are not influenced by the preference loss, resulting in a latent representation that lacks explicit correlation with average velocity. Consequently, the average velocity on the test set does not exhibit a monotonic pattern across all agents, and the velocity range for different latent values is limited (Table 5-1). In some other minibatches from the test set, the average velocity fails to follow a monotonic pattern even for individual agents. In contrast, a model trained with our PrefCAVE approach (Figure 5-1(b)) generates predictions where the average velocity increases monotonically with respect to z traversal. Specifically, we train the model such that larger latent values correspond to higher average velocities, resulting in a more distinct monotonic pattern and a broader velocity span on the test set. Interestingly, although the latent factors are regularized jointly during training, manipulating only one agent's latent factor (Figure 5-1(c)) while randomly sampling for all other agents in a scene effectively alters the behavior of that individual agent alone. This indicates that the latent factors are essentially learned independently for each agent.

Latent monotony consistency persists while moderately dropping random preferences. In pursuit of the goal of extending to human-labeled data, providing preferences for the entire dataset is costly. Therefore, it is important to examine the impact of randomly dropping preference pairs. Interestingly, applying preference pairs to the entire dataset ($\nu = 1$) does not always yield the best performance, which is counterintuitive. To evaluate this, we propose a benchmark called the violation rate (VR) to measure the consistency of monotony across the entire dataset. A violation occurs for an agent if, given two predictions $\hat{y}_0$ and $\hat{y}_1$ with latent generative factors $z_0 > z_1$, their average velocities satisfy $\text{avg\_vel}(x, \hat{y}_0) < \text{avg\_vel}(x, \hat{y}_1)$, where $z_i \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The test set consists of 138 scenes, encompassing 3,076 minibatches (each scene is recurrently clipped into 20-frame-long minibatches, averaging about 22 valid minibatches per scene) and 9,041 agents (with 1 to 11 agents per minibatch). The violation rates of monotony are calculated in three respects: agent-wise (if any trajectory violates), minibatch-wise (if at least one trajectory in the minibatch violates), and scene-wise (if at least one minibatch in the scene violates). Table 5-2 demonstrates that, for most configurations, we achieve a satisfactory violation rate (agent-wise VR smaller than 0.5%), and dropping preference pairs does not significantly increase violation rates. In fact, at a use rate of 25%, we achieve the optimal violation rate with comparable accuracy.

Leveraging only a partial set of preference pairs (less than 50% of the total) proves to be more effective because it reduces overfitting. While it is desirable to have as many high-quality preference pairs as possible, the model may also learn domain-specific noise present in these pairs, which can reduce its sensitivity to the actual semantic factors that the preference loss is intended to capture—in this case, the average velocity.

PrefCVAE encoder perceives more accurate characteristics of trajectories. We evaluate the encoder using CVAE-generated data as the ground truth. Following the traversal scheme described in the previous section, we assign values from 0.1 to 0.9 to the semantic latent factor, generating nine sets of predictions, $\hat{\mathbf{y}}_i$, each corresponding to one of the nine z

Table 5-2: Violation rate vs. preference use rate with different preference weight factors. Total latent dimension: 2, nuScenes dataset, manipulated scemantic factor is the average predicted velocity. (**SW VR**: Scene-wise violation rate; **MBW VR**: Minibatch-wise violation rate; **AW VR**: Agent-wise violation rate. A violation occurs when the factor is not monotonous w.r.t. z traversal. *All VR's: Lower is better*)

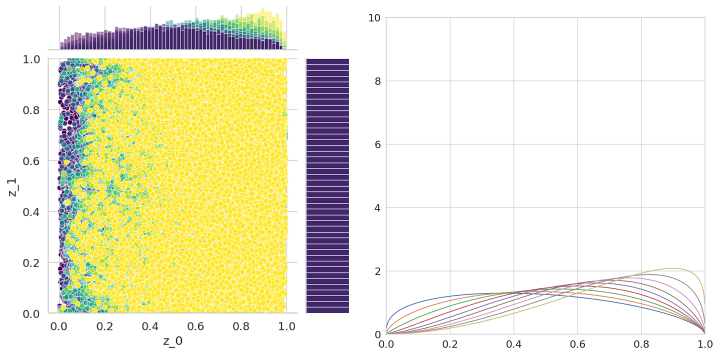
	Use rate, ν								
	0 (Base)	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
SW VR (%)	99.28	55.80	**4.35**	10.87	15.22	11.59	10.87	10.87	6.52
MBW VR (%)	92.43	12.55	**0.29**	0.85	0.91	0.65	0.68	0.75	0.35
AW VR (%)	82.28	4.86	**0.10**	0.29	0.32	0.22	0.23	0.25	0.12
minADE_5 (m)	2.62	2.54	2.66	2.58	2.64	2.59	2.71	2.53	2.64
minFDE_5 (m)	5.62	5.53	5.76	5.57	5.73	5.76	5.72	5.51	5.74

Table 5-3: Concentrated approximate posterior encoding with PrefCVAE and the resampling trick (**JSD**: Jensen-Shannon Divergence; L_{Mode}: The marginalized likelihood over all traversed latent values; *Avg. JSD and* $\log L_{\textit{Mode}}$*: Larger is better; Avg. Mode Dev.: Smaller is better*)

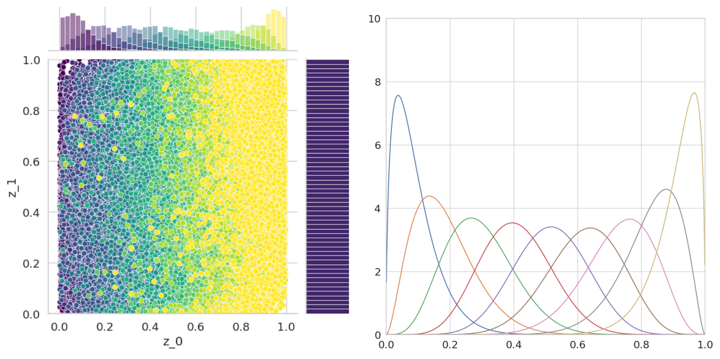
	Base	ν=0.25	ν=1
Avg. JSD	0.0352	**0.4874**	0.4578
$\log L_{\text{Mode}}$	3.19	**13.23**	11.76
Avg. Mode Dev.	0.1660	**0.0090**	0.0163

values used as the pseudo ground truth latent labels. For each set, we use the approximate posterior encoder to map the predictions back to the latent space, $\hat{\text{z}} \sim q_{\phi_i}(\mathbf{x}, \mathbf{y}_i)$. We then analyze the distribution of these nine sets of encoded latents. As shown in Figure 5-2 (first row, with the distribution on the x-axis representing the concerned latent dimension), the posterior encoder approximately encodes a Beta distribution for all agents in the test set. We use maximum likelihood estimation to fit a Beta distribution, $\hat{q}_{\phi_i}$, indicating the statistical pattern associated with each assigned z_i.

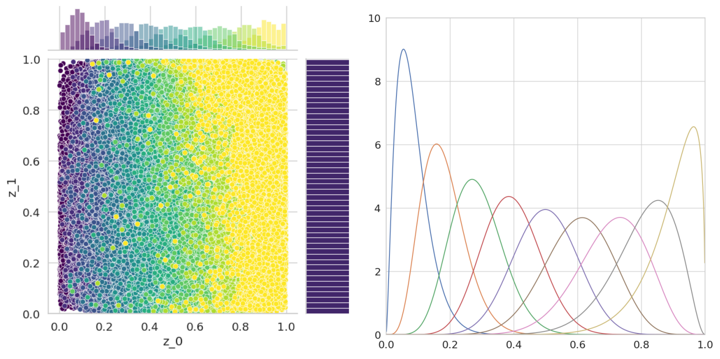
To assess the quality of the encoders, i.e., their semantic understanding ability, we adapt three metrics (Table 5-3). (i) The Jensen-Shannon divergence (JSD) measures the similarity between two distributions, and the average JSD, $\bar{D}_{\text{JS}}(q_{\phi_i}, q_{\phi_j})$ (calculated as the average of $C(9, 2) = 36$ pairwise JSDs for integer indices $i \neq j \in [1, 9]$), serves as a symmetric metric for the similarity of the nine distributions. A higher average JSD indicates greater dissimilarity between the encoded distributions. (ii) The log-likelihood of each distribution at its ground truth mode indicates how likely the ground truth is to be obtained. A higher cumulative log-likelihood, $\sum_{i=1}^{9} \log q_{\phi_i}(\text{z} = \frac{i}{10}; \mathbf{x}, \mathbf{y}_i)$, suggests that the latent distributions are more trustworthy. (iii) The mode deviation, $|\arg\max(q_{\phi_i}) - \frac{i}{10}|$, measures the deviation of the mode (the z value corresponding to the peak probability density) of each fitted Beta distribution from the ground truth. These three metrics collectively evaluate the distinction between different distributions, the concentration, and the estimation error of each distribution.



(a) Base



(b) ν=0.25



(c) ν=1

Figure 5-2: Distribution histograms and regressions of each traversed latent mode. Each colored histogram and regressed distribution pertain to trajectories predicted with a differently assigned z value. The ideal result should be 9 distinct distributions with modes at the ground truth z values. (Left: Marginal histogram; Right: Regressed beta distributions

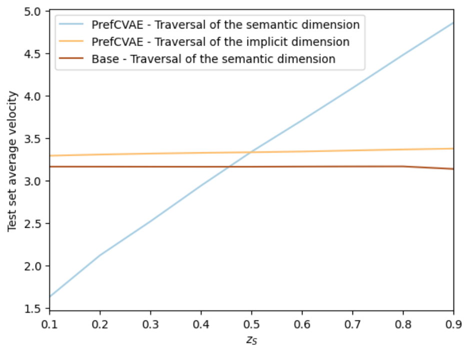


Figure 5-3: Velocity span as semantic latent traversal for different models and test time settings.

Table 5-4: Effect of tuning weighting factor at small preference use rates

Use rate (ν)	Violation rate type					
	Scene-wise (%)		Minibatch-wise (%)		Agent-wise (%)	
	λ=8	λ=32	λ=8	λ=32	λ=8	λ=32
0.05	**91.30**	96.38	64.56	**64.11**	45.99	**38.70**
0.10	93.47	**10.14**	65.08	**0.62**	46.09	**0.21**
0.15	91.30	**13.04**	63.30	**1.07**	44.62	**0.38**
0.20	95.65	**11.59**	77.44	**0.94**	56.89	**0.32**
0.25	38.41	**15.22**	6.05	**0.94**	2.18	**0.32**
0.30	84.06	**57.24**	24.80	**21.52**	**10.30**	11.04

5-3 Ablation Analysis

Manipulating unsupervised latent factors does not lead to explanatory prediction. Figure 5-3 demonstrates that when traversing the other latent dimension of the PrefCVAE-trained β-AgentFormer, the monotonic relationship with velocity does not persist. This indicates that our preference loss specifically supervises the assigned latent dimension and does not influence other dimensions.

Increasing the preference weighting factor λ enhances monotony consistency robustness. All previous experiments utilized a weighting factor of 16. We further explore the effects of setting λ to 8 and 32. As the previous section demonstrated that with $\lambda = 16$, a satisfactory violation rate (VR) can be achieved when ν exceeds 0.25, we focus on testing $\nu \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$ to avoid insignificant comparisons for $\nu > 0.3$. Due to the limited number of restarts (three different random seeds, as described in Section 5-1, with identical random seeds used for each ν), most models with $\lambda = 8$ fail to achieve a satisfactory VR (Table 5-4). Conversely, models with $\lambda = 32$ consistently achieve a very low VR, indicating that moderately increasing the weight of the PrefCVAE loss improves the model's robustness in understanding preferences. Furthermore, tuning the weighting factor does not substantially impact accuracy (Table 5-5). Therefore, we conclude that increasing the weighting factor can enhance explainability without compromising accuracy, although we have not determined an upper bound for an appropriate λ.

Table 5-5: Marginal effect on accuracy while tuning weighting factor at small preference use rates

Use rate (ν)	Accuracy			
	minADE$_5$ (m)		minFDE$_5$ (m)	
	λ=8	λ=32	λ=8	λ=32
0.05	2.59	2.52	5.57	5.52
0.10	2.57	2.52	5.59	5.46
0.15	2.53	2.57	5.48	5.55
0.20	2.73	2.51	5.97	5.52
0.25	2.53	2.54	5.52	5.55
0.30	2.58	2.71	5.56	5.91

Table 5-6: Emergence of tradeoff between accuracy and diversity as latent dimension increases

	Latent Dimension					
	nz=8		nz=16		nz=32	
	λ=16	λ=32	λ=16	λ=32	λ=16	λ=32
SW VR (%)	3.62	0.72	5.07	**0**	0.72	5.80
MNW VR (%)	0.20	0.03	0.23	**0**	0.03	0.26
AW VR (%)	0.07	0.01	0.08	**0**	0.01	0.09
Vel. Span (m/s)	1.74	**1.55**	1.74	**1.72**	1.80	1.68
(min/max)	5.03	**6.00**	5.08	**5.77**	4.96	4.91
minADE$_5$ (m)	2.70	2.83	2.70	2.83	**2.63**	**2.66**
minFDE$_5$ (m)	5.75	5.91	5.73	5.91	**5.57**	**5.61**

Increasing latent dimension nz introduces a tradeoff between accuracy and diversity. All previous experiments used a latent dimension of 2, with one dimension being semantic and the other domain-invariant. Having validated this concept, we now explore higher-dimensional latent spaces, which are more commonly used in prior works [3], [4]. Specifically, we investigate latent dimensions of 8, 16, and 32. For this set of comparisons, we use only the random seed 42, so some variance may exist. We find that increasing the latent dimension slightly improves the model's best accuracy (Table 5-6), likely because a larger information bottleneck can capture more implicit information about the trajectory history. However, the velocity span during traversal of the semantic latent space noticeably decreases. For nz = 8 or 16 with $\lambda = 32$, the maximum difference in average velocity exceeds 4.5 m/s, whereas for nz = 32, the span drops to below 3.3 m/s (as a baseline, the span for nz = 2 and $\lambda = 16$ is around 3.3 m/s), indicating reduced diversity in manipulation. This reduction occurs because, as the latent dimension increases, it becomes more challenging to enforce independence among latent factors, and the correlations between these dimensions make it harder to assign distinct semantics to each latent factor. Incorporating unsupervised disentanglement techniques could help mitigate this tradeoff [29], [31].

Chapter 6

Discussion

6-1 Limitations and Future Work

We identify two key assumptions underlying our work. (i) First, we use a simple oracle-based metric, the average prediction velocity, as the semantic factor. This represents a basic case study of our framework. For practical applications in autonomous driving, additional techniques are needed to incorporate human labeling and inject more practically useful semantics, such as Social Value Orientation (SVO) [45], [46]. (ii) Second, we only evaluate the effectiveness of our method with *one* semantic dimension. Reducing correlations among different latent factors is inherently challenging [31], making it difficult to extend our method to multiple semantic latent factors. Below, we propose tentative resolutions to these limitations.

High-level semantics. Our PrefCVAE framework can be extended to more complex semantic factors beyond the basic example of average velocity. Specifically, SVO is a valuable indicator of driver behavior [46]. Since SVO for a trajectory can be estimated in a closed-form manner given a predefined or hand-crafted reward model, the pipeline introduced in this paper can be directly extended to learn SVO as a latent factor.

Adaptation to human-labeled preferences. Although our preference loss does not explicitly use the ground truth of metric values (the Sigmoid function output only indicates the relationship between a pair), the preferences we calculate are still based on ground truth metric calculations. This is feasible because average velocity is easily computed. However, in practice, we may require more abstract semantic factors that are difficult to define. As future work, we aim to relax the assumption of ground truth requirements in human labeling scenarios. This can be achieved using a Straight-Through Estimator (STE) [47] or Gumbel-Softmax [48] to approximate the preference. The key challenge is that human-labeled preference data is not inherently differentiable, but STE or Gumbel-Softmax can address such samples. Additionally, it may be worthwhile to first fit a neural network-based preference classifier and then use it to generate synthesized human-like preferences.

Multiple semantic representations. In this work, we only investigate the effectiveness of learning *one* metric at a time. However, the promise of disentangled representation learning lies in the ability to learn a set of generative factors, rather than just one. Therefore, it is worth exploring the incorporation of multiple metrics as different latent dimensions. Previous efforts in disentanglement research could be beneficial in addressing these challenges [49].

6-2 Implications for Other Machine Learning Tasks

Explanatory trajectory prediction using CVAE is just one application of the PrefCVAE framework. We tentatively explore its utility in generic imitation learning and data generation tasks.

Imitation learning. Imitation learning involves learning actions from offline demonstrations. Generally, there are two approaches: behavior cloning and inverse reinforcement learning [50]. Robotics tasks learned with generative models often contain biases from the dataset. A recent work, SkiP [51], proposed using a VAE-based model to extract skill priors with human preference, followed by RLHF to fine-tune the generative policy. Our work offers new insights into this area: rather than using an external classifier to approximate human preference, we demonstrate that the approximate posterior encoder of a CVAE can inherently function as a preference estimator.

Data generation. Generating abundant and authentic training data is increasingly important for large-scale pretraining with foundation models [52]. Using VAE-based models to generate such data is one possible approach [53]. Our work suggests a method to introduce bias into the data generation process, aligning the dataset distribution with human preferences.

Chapter 7

Conclusion

This paper presents ***PrefCVAE***, an augmentation to the CVAE framework that enables explanatory trajectory prediction. The core innovation is a preference loss that regularizes the semantic meanings of latent factors through pairwise preference comparison. Such comparisons are often tractable and cost-effective for many tasks. Beyond proposing a method for explanatory trajectory prediction, we aim to offer a new perspective on effectively and efficiently incorporating dataset inductive bias for disentangled representation learning in deep generative models.

Acknowledgements

First and foremost, biggest thanks to my parent and other family members, who have supported me, both spiritually and financially, over the past course of my studies. You holding my back is essential for me to continue striving on the journey of research.

I also greatly appreciate the support and supervision from Arkady and Julian. This project cannot be concluded without your countless helps. You have also trained me to be a better researcher in many ways.

Lastly, to my friends. Thank you all for being in this chapter of my life.

Yongxi Cao
TU Delft, the Netherlands
August 23, 2024

Appendix A

Conditional Variational Autoencoders

Probabilistic graph. A wide range of machine learning tasks such as classification or synthesizing using deep probabilistic generative models assume different dependency relationships among random variables. Specifically, the notation in trajectory prediction differs from normal vision works. In this section we provide a probabilistic view of the multiple variables to clarify dependencies.

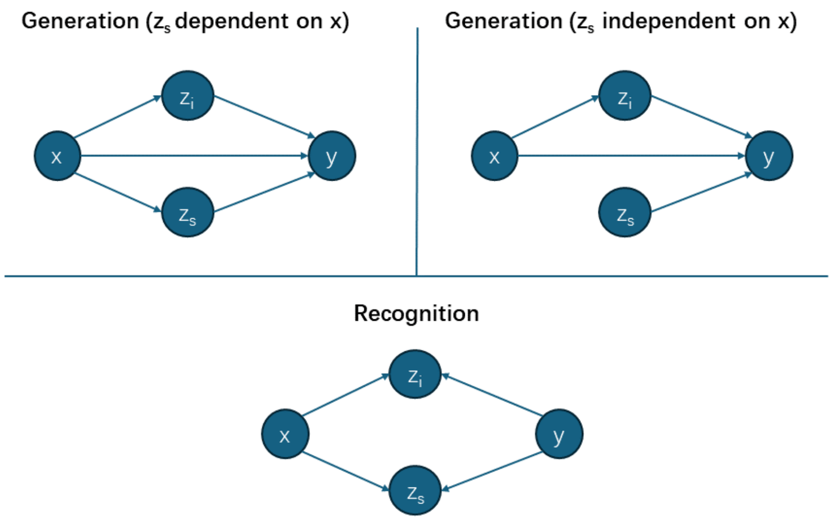
The following random variables are considered,

- $\mathbf{x}$: History observation
- $\mathbf{y}$: Future trajectory
- $\mathbf{z}_i$: Implicit latent generative factors, who are not explicitly learned with our proposed loss
- $\mathbf{z}_s$: Semantic latent generative factors, who are explicitly learned with our proposed loss

Given the definition of random variables, the probabilistic graph for generation of prediction and recognition of entire trajectories are shown in Figure.A-1.

The two types of generative relationships represent two plausible assumptions: (i). Some factors of future trajectory should be consistent with history and thus can be inferred from the history. For example, driving style related factors should be consistent throughout the entire trajectory, and be able to be estimated based on history observation. (ii). On the other hand, some characters could be considered independent. For example, the choice of future velocity could be diverse even with the identical history observation, with all candidate predictions being plausible.

The major difference in the causal framework between the two cases occurs in the assumption of prior distribution. For the former case, the prior distribution is assumed to be learnable, meaning that the prior can be variational conditioned on the history observation; For the



Generation (z_s dependent on x)
Generation (z_s independent on x)
Recognition
z_i
x
y
z_s

Figure A-1: Probabilistic graph for generation and recognition

latter case, the prior should be unlearnable, meaning that no prior knowledge regarding this generative factor could be drawn from the history observation.

ELBO loss deduction. To learn a generative multimodal conditional distribution, a conditional variational autoencoder (CVAE) assumes a marginalized distribution

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \int p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) p_\theta(\boldsymbol{z}|\boldsymbol{x}) dz. \tag{A-1}$$

For ease of loss calculation for a batch of data, the log-likelihood of the conditional distribution $\log p_\theta(y|x)$ is often considered,

$$\begin{aligned}
\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) &= \log \int p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) p_\theta(\boldsymbol{z}|\boldsymbol{x}) dz \\
&= \log \int \frac{p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) p_\theta(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})} q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) dz \quad \text{(Introduce approximate posterior)} \\
&= \log \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} \left[\frac{p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) p_\theta(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})} \right] \quad \text{(By definition of expectation)} \\
&\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} \left[\log \frac{p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) p_\theta(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})} \right] \quad \text{(Jensen's inequality)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} [\log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} \left[\log \frac{p_\theta(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})} \right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} [\log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})] - \mathbf{D}_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) || p_\theta(\boldsymbol{z}|\boldsymbol{x})). \quad \text{(Definition of KLD)}
\end{aligned} \tag{A-2}$$

In traditional Bayesian inference, the posterior is updated upon every likelihood observation. However, in CVAEs the probability distributions are assumed to be generalisable and parameterized by neural networks, making it intractable to calculate. Hence, it is common practice to leverage another recognition distribution to approximate the true posterior, $q_\phi(z|x, y)$, which is often called approximate posterior. In the deduction, θ denotes the parameter of the prior distribution encoder and the likelihood distribution decoder, and ϕ denotes parameter of the approximate posterior distribution encoder.

In practice, the first term in ELBO loss is estimated with reparameterization trick through Monte-Carlo sampling the latent random variable, and the KL-divergence term is distribution-specific, and is close-formed for most common distributions.

Appendix B

Implementation Details

B-1 Training Details

The model is trained on an Ubuntu 20.04 workstation with an Intel® Core™ i9-11900KF @3.50GHZ CPU, 1 NVIDIA® GeForce® RTX 3090 GPU, and 64 GB memory. For the nuScenes dataset, a model with a total of 2-dimensional latent space and 1 semantic dimension takes around 24 hours to train for 50 epochs. For all parallel comparisons, we use the DAIC clusters at the TU Delft with A100 GPUs.

Important software stack versions: CUDA 11.7, PyTorch 1.13.1, CUDNN: 8.5.0. Please note that different versions of software may bring minuscule numerical differences from results listed in this report, although random seed and non-deterministic flags are preset to maximize reproducability (see also: https://pytorch.org/docs/stable/notes/randomness.html).

B-2 Task-specific Losses

We observed that, while using the original AgentFormer loss with our PrefCVAE loss term, the predictions typically has a very large deviation at the first time step, and for the rest of future trajectory are normal. Hence, we add a regulation to the first step prediction: $\mathcal{L}_{\text{task}} = ||\mathbf{Y}^{T_{\text{cur}+1}} - \hat{\mathbf{Y}}^{T_{\text{cur}+1}}||_2$, with a weighting factor of 8. To reduce effect of this additional term during comparison, we use it for *all* experiments presented in this paper.

B-3 Semantic Factors and Oracle Programs

The new preference loss takes the preference between two trajectories as the input, and the preference over two trajectories is given by an oracle program or human labeler. As validation of the concept, in this work we introduce two close-form oracle programs to provide such preference by firstly estimating each individual ground truth metric value and then comparing them. But one should note that such oracle could also be replaced by the preference given by human labelers. Also, these proxies are to validate the methodology, and more applicable alternatives could be beneficial to learning other useful latent factors. Lastly, we discuss how to further leverage human labels instead of oracle program in our framework.

We consider scemantics for an entire scene of a fixed number of N agents, with 4 history frames and 12 future frames. Then, the variables required for the oracle calculation are

- **History position observation:** $\mathbf{X} = [\{\boldsymbol{ph}_t\}_{t=0}^{3}] = [\{(xh_t, yh_t)\}_{t=0}^{3}]$
- **Future position prediction:** $\mathbf{Y} = [\{\boldsymbol{pf}_t\}_{t=0}^{11}] = [\{(xf_t, yf_t)\}_{t=0}^{11}]$. Also, denote $\boldsymbol{pf}_{-1}$ to be $\boldsymbol{ph}_3$
- **Future velocities:** $\mathbf{V}_Y = [\{\boldsymbol{vf}_t\}_{t=0}^{11}] = [\{(vxf_t, vyf_t)\}_{t=0}^{11}]$. Also, denote $\boldsymbol{vf}_{-1}$ to be $\boldsymbol{vh}_3$
- **Semantic map:** $\mathcal{M}$: Information of occupancy pertain to each position in the scene, indicating the ground type, such as drive lanes, pavement, etc.

When there are confusions in terms of the agent index in the scene, we apply a superscript to distinguish them.

B-3-1 Low-level: Average Velocity of Prediction

In the low level oracle, we define larger average velocity to pertain to larger latent factor values. The average velocity is simply calculated as

$$\mathbf{m}_{\text{avg\_vel}}(\mathbf{V}_Y) = \sum_{i=0}^{11} \|(vxf_t, vyf_t)\|_2 \tag{B-1}$$

Since velocity can be agent-independent, i.e., all agents could simultaneously be faster or slower, the velocity is taken average further over the entire mini-batch, i.e., the current scene with N agents.

B-3-2 High-level: Social Value Orientation

Similar to the low-level case, the oracle program also firstly estimates ground truth SVOs and formulates the result as the comparison between the pair. The oracle reward for a single agent involves several aspects, as described in existing literature.

Reward definition We define an oracle reward for each individual (larger reward value implies better) with following terms:

- **Collision avoidance, r_{colli}:** Model the reward function to penalize the agents for getting too close to each other.
- **Comfort, r_{comf}:** Quadratically penalizes high steering and acceleration.

- **Centered positions, $\boldsymbol{r}_{\text{cent}}$:** Penalizes the trajectory's deviation from its current lane's center line, which is regressed from semantic map.
- **Road departure, $\boldsymbol{r}_{\text{dept}}$:** Assigns a negative value on occurrence of the predicted trajectory misses derivable area, which is provided by the semantic map.

Denote the Euclidean distance between the ith agent and the jth agent at time t is defined as d_{ij}. To penalize closeness, we use a logarithmic barrier function that becomes increasingly large as the distance $d_{ij}(t)$ approaches a predefined safe distance d_{safe}

$$B(d_{ij}(t)) = -\log\left(\frac{d_{ij}(t)}{d_{\text{safe}}}\right)$$

The function $B(d_{ij}(t))$ is active when $d_{ij}(t) < d_{\text{safe}}$.

Then the collision avoidance reward $R_i(t)$ for the ith agent, considering all other agents in the scene, is defined as

$$\boldsymbol{r}_{\text{colli}}(Y) = \sum_{t=0}^{11} \sum_{j=1, j\neq i}^{N} \left[-\lambda \cdot \mathbb{I}\left(d_{ij}(t) < d_{\text{safe}}\right) \cdot B(d_{ij}(t))\right]$$

where:

- $\lambda > 0$ is a scaling factor that determines the severity of the penalty.
- $\mathbb{I}\left(d_{ij}(t) < d_{\text{safe}}\right)$ is an indicator function that is 1 if $d_{ij}(t) < d_{\text{safe}}$ and 0 otherwise.

The centered positions term is defined as

$$\boldsymbol{r}_{\text{comf}}(Y) = \sum_{t=0}^{11} \cos^2(\varphi(\boldsymbol{vf}_t, \boldsymbol{vf}_{t-1})) - \|(vxf_t, vyf_t)\|_2^2, \tag{B-2}$$

where φ is the angle between two frames' velocities indicating the steering,

$$\cos(\varphi(\boldsymbol{vf}_t, \boldsymbol{vf}_{t-1})) = \frac{< \boldsymbol{vf}_t, \boldsymbol{vf}_{t-1} >}{\|\boldsymbol{vf}_t\| \cdot \|\boldsymbol{vf}_{t-1}\|}. \tag{B-3}$$

Assume the regressed equation of lane center line closest to the predicted trajectory is given by

$$Ax + By + C = 0. \tag{B-4}$$

Then the reward on centered positions is the negative sum of deviation of the trajectory,

$$\boldsymbol{r}_{\text{cent}}(Y) = -\sum_{t=0}^{11} \frac{|Axf_t + Byf_t + C|}{\sqrt{A^2 + B^2}} \tag{B-5}$$

In practice, for the NuScenes dataset, the distances are not calculated with the theoretical formula above, but with the map API, which gives the closest pose on the closest lane from the queried pose. Then the Euclidean distance can be easily calculated.

Since there are the cases of straight line or a curvature exists, we apply a coarse approximation to the linear equation for estimation of lane centerline is simply regressed from the closest 3 points on the lane. The closest point on a lane can be queried given a pose, and the adjacent two points are also used.

Now that the semantic map is available, given every predicted position, the semantic could be queried. Then road departure is penalized when the prediction misses drivable area,

$$\boldsymbol{r}_{\text{dept}}(Y, \mathcal{M}) = \begin{cases} -R_{\text{dept}}, & \text{if } Y \text{misses drivable area in } \mathcal{M}, \\ 0 & \text{otherwise,} \end{cases} \tag{B-6}$$

where R_{dept} is a constant.

Then the total reward for agent i is

$$\boldsymbol{r}^i = \boldsymbol{r}^i(\boldsymbol{Y}, \boldsymbol{V}_Y, \mathcal{M}) = w_0 \boldsymbol{r}_{\text{colli}} w_1 \boldsymbol{r}_{\text{comf}} + w_2 \boldsymbol{r}_{\text{cent}} + w_3 \boldsymbol{r}_{\text{dept}}, \tag{B-7}$$

where w_0, w_1, w_2 and w_3 are constant weights.

For the i-th agent among the N agents in the scene with SVO of ϕ_i, the cumulative scene reward is

$$\boldsymbol{r}_{\text{total,i}}(\phi_i; \boldsymbol{Y}, \boldsymbol{V}_Y, \mathcal{M}) = \cos(\phi_i)\boldsymbol{r}^i + \sin(\phi_i)\frac{1}{N-1}\sum_{\substack{j=1 \\ j \neq i}}^{N} \boldsymbol{r}^j, \tag{B-8}$$

where $\boldsymbol{Y} = \left[Y^0, ..., Y^{N-1}\right]$.

SVO Estimation Given the agent-wise cumulative reward defined above, the close-form optimum of the SVO is given by

$$\boldsymbol{\phi}_i^* = \underset{\phi \in (-\frac{\pi}{2}, \frac{\pi}{2})}{\arg\max} \left(\boldsymbol{r}_{\text{total,i}}(\phi_i; \boldsymbol{Y}, \boldsymbol{V}_Y, \mathcal{M})\right) = \text{arctan2}\left(-\frac{\partial \boldsymbol{r}^i(\boldsymbol{Y}, \boldsymbol{V}_Y, \mathcal{M})}{\partial Y^i}, \frac{\partial \sum_{\substack{j=1 \\ j \neq i}}^{N} \boldsymbol{r}^j(\boldsymbol{Y}, \boldsymbol{V}_Y, \mathcal{M})}{(N-1)\partial Y^i}\right). \tag{B-9}$$

Then for the minibatch, the high-level metric is the vectorized SVOs

$$\boldsymbol{m}_{\text{SVO}}(\boldsymbol{Y}, \boldsymbol{V}_Y, \mathcal{M}) = \left[\boldsymbol{\phi}_0^*, \boldsymbol{\phi}_1^*, ..., \boldsymbol{\phi}_{N-1}^*\right]. \tag{B-10}$$

B-4 Latent Distributions

In this section we describe the probability distribution, KL-divergence calculation, and reparameterization trick for both levels of semantics.

Unlike in the base models where Gaussian distributions is applied to model the latent distribution, we choose Beta and von Mises-Fisher distributions to cater to characteristics of estimation of bounded velocity and SVO. For CVAEs, the prior and approximate posterior encoders encode the data to distribution parameters, and the latent factors are sampled from the probability distribution defined by these parameters. For different levels of semantics, the outer probabilistic framework is generalizable. Here we describe the probability distribution definitions, their KL-divergence calculation, and how reparameterization tricks enables gradient to flow through them.

B-4-1 Beta Distribution

Probability density function A Beta distribution has a bounded domain of $z \in [0, 1]$ and two governing parameters α and β and is defined as

$$\mathbf{Beta}(z|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} z^{\alpha-1}(1-z)^{\beta-1}, \tag{B-11}$$

where $B(\alpha, \beta)$ is the Beta function

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\, dt, \tag{B-12}$$

and $\Gamma(x)$ is the Gamma function

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt \tag{B-13}$$

In our model, we desire the latent Beta distribution to possess a mode within the range of $(0, 1)$, so α and β are clipped to be larger than 1. This is enforced by adding an intercepted exponential linear unit (IcpELU) layer after the encoder, which is defined as

$$\text{IcpELU}(x) = \begin{cases} x+2, & \text{if } x > 0, \\ \exp(x)+1 & \text{if } x \leq 0. \end{cases} \tag{B-14}$$

That is, for $(a, b) = \text{PriorProj}(\text{AFEncoder}(X))$ or $(a, b) = \text{PostProj}(\text{AFEncoder}(X, Y))$, $(\alpha, \beta) = (\text{IcpELU}(a), \text{IcpELU}(b))$.

There is two-fold advantage of using Beta distribution instead of Gaussian as average velocity indicator: (i). Uniform distribution, used in unlearnable prior distribution case, is essentially Beta distribution with $\alpha = \beta = 1$, so the KL divergence computation when the prior is uniform distribution brings no extra trouble; (ii). No extra process is required to regulate the output of the Gaussian distribution to a fixed range, like [0,1] in our design, since the Beta is inherently a bounded distribution.

KL Divergence The KL divergence between two Beta distributions $\text{Beta}(\alpha_1, \beta_1)$ and $\text{Beta}(\alpha_2, \beta_2)$ is given by

$$\begin{aligned} D_{KL}(\text{Beta}(\alpha_1, \beta_1) \parallel \text{Beta}(\alpha_2, \beta_2)) = \log \frac{B(\alpha_2, \beta_2)}{B(\alpha_1, \beta_1)} + (\alpha_1 - \alpha_2)\psi(\alpha_1) \\ + (\beta_1 - \beta_2)\psi(\beta_1) + (\alpha_2 - \alpha_1 + \beta_2 - \beta_1)\psi(\alpha_1 + \beta_1) \end{aligned} \tag{B-15}$$

where ψ is the digamma function, which is the derivative of the logarithm of the gamma function

$$\psi(x) = \frac{d}{dx} \log \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}. \tag{B-16}$$

Reparameterization trick A random variable X following a beta distribution, denoted $X \sim \text{Beta}(\alpha, \beta)$, cannot be differentiated through in a standard backpropagation. To reparametrize, the process involves the following steps: The beta distribution can be generated from two independent gamma distributions. Specifically, if $Y_1 \sim \text{Gamma}(\alpha, 1)$ and $Y_2 \sim \text{Gamma}(\beta, 1)$, then

$$X = \frac{Y_1}{Y_1 + Y_2}$$

follows a beta distribution $X \sim \text{Beta}(\alpha, \beta)$.

The gamma distribution with shape parameter k and scale 1 can be reparametrized using a standard gamma distribution with shape parameter 1 (which is equivalent to an exponential distribution). For a sample $Z \sim \text{Gamma}(1, 1)$, the sample from a gamma distribution with shape k can be obtained by:

$$Y = G(k) = \sum_{i=1}^{k} Z_i$$

where $Z_i \sim \text{Exp}(1)$ are independent samples from an exponential distribution.

A more practical approach is to approximate the gamma distribution using existing reparametrization techniques, such as the inverse of the cumulative distribution function (CDF) method or leveraging approximations for more complex cases.

Once Y_1 and Y_2 are obtained through reparametrization, the beta-distributed variable X can be constructed as:

$$X = \frac{G(\alpha)}{G(\alpha) + G(\beta)}$$

This expression now allows gradients to flow back through the parameters α and β because the operations involved are differentiable.

B-4-2 Von Mises Distribution

The von Mises distribution, often referred to as the circular normal distribution, is a probability distribution on the circle. In 2D, it is used to model angular data and is defined by a mean direction vector $\boldsymbol{\mu}$ and a concentration parameter κ.

Probability density function. The probability density function of a 2D von Mises distribution is given by:

$$\mathrm{VM}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp\left(\kappa \boldsymbol{\mu}^\top \boldsymbol{\theta}\right)$$

where:

- $\boldsymbol{\theta}$ is a unit vector on the circle.
- $\boldsymbol{\mu}$ is the mean direction vector (also a unit vector).
- $\kappa \geq 0$ is the concentration parameter, with larger values of κ indicating stronger concentration around the mean direction.
- $I_0(\kappa)$ is the modified Bessel function of the first kind and order zero, which serves as a normalization constant.

KL Divergence. The Kullback-Leibler divergence between two 2D von Mises distributions $\mathrm{VM}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_1, \kappa_1)$ and $\mathrm{VM}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_2, \kappa_2)$ is given by:

$$D_{KL}(\mathrm{VM}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_1, \mathrm{VM}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_2, \kappa_2)) = \log\left(\frac{I_0(\kappa_2)}{I_0(\kappa_1)}\right) + (\kappa_1 \boldsymbol{\mu}_1^\top \boldsymbol{\mu}_2 - \kappa_2)$$

This expression measures the divergence or "distance" between the two distributions.

Reparameterization Trick. In variational inference, particularly in variational autoencoders (VAEs), the reparameterization trick is used to allow gradients to flow through stochastic nodes. For the 2D von Mises distribution, the reparameterization trick can be applied as follows:

- First, sample a unit vector $\boldsymbol{z}$ from a uniform distribution over the circle.
- Then, apply the following transformation to obtain a sample $\boldsymbol{\theta}$ from the von Mises distribution:

  $$\boldsymbol{\theta} = \frac{\boldsymbol{\mu} + \sqrt{1 - \left(\frac{\kappa}{\|\boldsymbol{\mu}\|}\right)^2}\, \boldsymbol{z}}{\|\boldsymbol{\mu} + \sqrt{1 - \left(\frac{\kappa}{\|\boldsymbol{\mu}\|}\right)^2}\, \boldsymbol{z}\|}$$

  Here, $\boldsymbol{z}$ is independent of the parameters $\boldsymbol{\mu}$ and κ, which allows backpropagation through the sampling process.

This reparameterization allows the von Mises distribution to be used effectively in gradient-based optimization, enabling the incorporation of angular stochasticity in neural networks.

References

[1] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[2] F. S. Westerhout, J. F. Schumann, and A. Zgonnikov, "Smooth-trajectron++: Augmenting the trajectron++ behaviour prediction model with smooth attention," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2023, pp. 5423–5428.

[3] 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.

[4] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9813–9823.

[5] X. Huang, S. G. McGill, J. A. DeCastro, *et al.*, "Diversitygan: Diversity-aware vehicle motion prediction via latent semantic sampling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5089–5096, 2020.

[6] P. Dendorfer, S. Elflein, and L. Leal-Taixé, "Mg-gan: A multi-generator model preventing out-of-distribution samples in pedestrian trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 158–13 167.

[7] Y. Choi, R. C. Mercurius, S. M. A. Shabestary, and A. Rasouli, "Dice: Diverse diffusion model with scoring for trajectory prediction," *arXiv preprint arXiv:2310.14570*, 2023.

[8] 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.

[9] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," *Advances in neural information processing systems*, vol. 27, 2014.

[10] B. Ivanovic and M. Pavone, "Injecting planning-awareness into prediction and detection evaluation," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2022, pp. 821–828.

[11] M. Itkina and M. Kochenderfer, "Interpretable self-aware neural networks for robust trajectory prediction," in *Conference on Robot Learning*, PMLR, 2023, pp. 606–617.

[12] Y. Liu, R. Cadei, J. Schweizer, S. Bahmani, and A. Alahi, "Towards robust and adaptive motion forecasting: A causal representation perspective," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 081–17 092.

[13] Q. Zhang, T. Zhang, and L. Ma, "Human acceptance of autonomous vehicles: Research status and prospects," *International journal of industrial ergonomics*, vol. 95, p. 103 458, 2023.

[14] W. Zeng, W. Luo, S. Suo, *et al.*, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[15] S. Hagedorn, M. Hallgarten, M. Stoll, and A. Condurache, "Rethinking integration of prediction and planning in deep learning-based automated driving systems: A review," *arXiv preprint arXiv:2308.05731*, 2023.

[16] X. Huang, G. Rosman, A. Jasour, S. G. McGill, J. J. Leonard, and B. C. Williams, "Tip: Task-informed motion prediction for intelligent vehicles," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 11 432–11 439.

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, "Parameter-efficient transfer learning for nlp," in *International conference on machine learning*, PMLR, 2019, pp. 2790–2799.

[18] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[19] P. Kothari, D. Li, Y. Liu, and A. Alahi, "Motion style transfer: Modular low-rank adaptation for deep motion forecasting," in *Conference on Robot Learning*, PMLR, 2023, pp. 774–784.

[20] J. Wang, K. Messaoud, Y. Liu, J. Gall, and A. Alahi, "Forecast-peft: Parameter-efficient fine-tuning for pre-trained motion forecasting models," *arXiv preprint arXiv:2407.19564*, 2024.

[21] Y.-L. Kuo, X. Huang, A. Barbu, *et al.*, "Trajectory prediction with linguistic representations," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2868–2875.

[22] T.-H. Wang, A. Maalouf, W. Xiao, *et al.*, "Drive anywhere: Generalizable end-to-end autonomous driving with multi-modal foundation models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 6687–6694.

[23] I. Bae, J. Lee, and H.-G. Jeon, "Can language beat numerical regression? language-based multimodal trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 753–766.

[24] S. Sreeram, T.-H. Wang, A. Maalouf, G. Rosman, S. Karaman, and D. Rus, "Probing multimodal llms as world models for driving," *arXiv preprint arXiv:2405.05956*, 2024.

[25] I. Higgins, L. Matthey, A. Pal, *et al.*, "Beta-vae: Learning basic visual concepts with a constrained variational framework.," *ICLR (Poster)*, vol. 3, 2017.

[26] A. Goyal and Y. Bengio, "Inductive biases for deep learning of higher-level cognition," *Proceedings of the Royal Society A*, vol. 478, no. 2266, p. 20 210 068, 2022.

[27] B. Paige, J.-W. Van De Meent, A. Desmaison, *et al.*, "Learning disentangled representations with semi-supervised deep generative models," *Advances in neural information processing systems*, vol. 30, 2017.

[28] F. Locatello, S. Bauer, M. Lucic, *et al.*, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *international conference on machine learning*, PMLR, 2019, pp. 4114–4124.

[29] D. Zietlow, M. Rolinek, and G. Martius, "Demystifying inductive biases for (beta-) vae based architectures," in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 945–12 954.

[30] H. Kim and A. Mnih, "Disentangling by factorising," in *International conference on machine learning*, PMLR, 2018, pp. 2649–2658.

[31] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," *Advances in neural information processing systems*, vol. 31, 2018.

[32] F. Locatello, B. Poole, G. Rätsch, B. Schölkopf, O. Bachem, and M. Tschannen, "Weakly-supervised disentanglement without compromises," in *International conference on machine learning*, PMLR, 2020, pp. 6348–6359.

[33] L. Lee, B. Eysenbach, R. R. Salakhutdinov, S. S. Gu, and C. Finn, "Weakly-supervised reinforcement learning for controllable behavior," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2661–2673, 2020.

[34] X. Shen, F. Liu, H. Dong, Q. Lian, Z. Chen, and T. Zhang, "Weakly supervised disentangled generative causal representation learning," *Journal of Machine Learning Research*, vol. 23, no. 241, pp. 1–55, 2022.

[35] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of gans for semantic face editing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9243–9252.

[36] Z. Wu, D. Lischinski, and E. Shechtman, "Stylespace analysis: Disentangled controls for stylegan image generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 863–12 872.

[37] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[38] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.

[39] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[40] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *Advances in neural information processing systems*, vol. 31, 2018.

[41] D. M. Ziegler, N. Stiennon, J. Wu, *et al.*, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[42] N. Stiennon, L. Ouyang, J. Wu, *et al.*, "Learning to summarize with human feedback," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[43] L. Ouyang, J. Wu, X. Jiang, *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[44] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[45] R. O. Murphy, K. A. Ackermann, and M. J. Handgraaf, "Measuring social value orientation," *Judgment and Decision making*, vol. 6, no. 8, pp. 771–781, 2011.

[46] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[47] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[48] E. Jang, S. Gu, and B. Poole, "Categorical reparametrization with gumble-softmax," in *International Conference on Learning Representations (ICLR 2017)*, OpenReview. net, 2017.

[49] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, "Disentangling factors of variation using few labels," *arXiv preprint arXiv:1905.01258*, 2019.

[50] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[51] X. Wang, K. Lee, K. Hakhamaneshi, P. Abbeel, and M. Laskin, "Skill preferences: Learning to extract and execute robotic skills from human feedback," in *Conference on Robot Learning*, PMLR, 2022, pp. 1259–1268.

[52] J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh, and P. Das, "Large-scale chemical language representations capture molecular structure and properties," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1256–1264, 2022.

[53] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2901–2910.