A APPENDIX

A.1 TRAINING ALGORITHM OF ITPNET

We present the pesudo code of training ITPNet in Algorithm 1

Algorithm 1: Training Procedure of ITPNet

Input: input trajectory $\mathbf{X} = \{\mathbf{X}^{obs}, \mathbf{X}^{unobs}\}$, ground-truth trajectory $\mathbf{X}^{gt}$, query embedding $\mathbf{Q}$, layers L of NRRFormer, trade-off hyper-parameters: α, and β.
Output: Network parameters: ϕ, ψ, $\{\theta_{l,1}, \theta_{l,2}, \theta_{l,3}\}_{l=1}^{L}$, and ω.
Initialize: Randomly initialize ϕ, ψ, $\{\theta_{l,1}, \theta_{l,2}, \theta_{l,3}\}_{l=1}^{L}$, ω, and $\mathbf{Q}$.
while *not converges* **do**
- Compute latent feature representations $\mathbf{V}^{obs} = \Phi(\mathbf{X}^{obs}; \phi)$ and $\mathbf{V}^{unobs} = \Phi(\mathbf{X}^{unobs}; \phi)$;
- Backward forecast $\widehat{\mathbf{V}}^{unobs}$ by $\widehat{\mathbf{V}}^{unobs} = \Psi(\mathbf{V}^{obs}; \psi)$;
- Compute $\mathcal{L}_{rec}, \mathcal{L}_{cts}$ by Eq. (5) and (7), respectively;
- Employ NRRFormer to filter out redundancy and noise in predicted unobserved latent feature representations and integrate the resulting filtered feature representations and observed feature representations into Q, by
- $\hat{\mathbf{V}}_0^{unobs} = \hat{\mathbf{V}}^{unobs}$;
- **for** $l = 0...L-1$ **do**
  - $\mathbf{Q}_l^{unobs}, \hat{\mathbf{V}}_{l+1}^{unobs} = \textbf{SelfAtt}(\mathbf{Q}_l || \hat{\mathbf{V}}_l^{unobs}; \theta_{l,1})$;
  - $\mathbf{Q}_l^{unobs,obs}, \mathbf{V}^{obs*} = \textbf{SelfAtt}(\mathbf{Q}_l^{unobs} || \mathbf{V}^{obs}; \theta_{l,2})$;
  - $\mathbf{Q}_{l+1} = \textbf{FeedForward}(\mathbf{Q}_l^{unobs,obs}; \theta_{l,3})$;
- **end**
- Predict trajectory $\{\widehat{\mathbf{X}}^k\}_{k\in[0,K-1]} = \Omega(\mathbf{Q}_L; \omega)$;
- Compute $\mathcal{L}_{reg}, \mathcal{L}_{cls}$ through $\{\widehat{\mathbf{X}}^k\}_{k\in[0,K-1]}$;
- Calculate the total loss $\mathcal{L}$ by $\mathcal{L} = \mathcal{L}_{reg} + \mathcal{L}_{cls} + \alpha\mathcal{L}_{rec} + \beta\mathcal{L}_{cts}$;
- Update model parameters ϕ, ψ, $\{\theta_{l,1}, \theta_{l,2}, \theta_{l,3}\}_{l=0}^{L-1}$, ω and query embedding $\mathbf{Q}$ by minimizing $\mathcal{L}$.

end

A.2 RESULTS WITH DIFFERENT LENGTHS OF OBSERVATIONS

To further demonstrate the effectiveness of our method, we perform HiVT and LaneGCN with different lengths of observed locations T. Table 4 reports the results. One interesting point is that our ITPNet+LaneGCN with $T = 2$ achieves comparable performance to LaneGCN with $T = 5$ when $K = 1$ on the nuScenes dataset. This means that our method can averagely save 1.5 seconds for trajectory prediction, compared to LaneGCN. If a car has a driving speed of 70 kilometers per hour on an urban road, our method can save around 30 meters to observe the agent for trajectory prediction, compared to LaneGCN.

A.3 CONVERGENCE ANALYSIS

We study the convergence of our method on Argoverse and nuScenes. The curves of the total loss of our method are shown in Figure 5. we can see the loss decreases as the training steps, and it finally levels off.

A.4 FAILURE CASES OF ITPNET

We provide failure cases of ITPNet+HiVT on Argoverse dataset, as shown in Figure 6. The model fails (1) when the future intention of the agents suddenly changes (a, d); (2) the future behavior is complex and hard to perceive from observed trajectories, such as overtaking; (3) the agent does not follow the traffic rules, such as turning left from the lane for right turns (c).

Table 4: minADE@K, minFDE@K, and MR@K of methods with different observed locations (T) on Argoverse and nuScenes, respectively.

Dataset	Method	T	K=1			K=6		
			minADE	minFDE	minMR	minADE	minFDE	minMR
Argoverse	HiVT (Zhou et al., 2022)	3	2.958	6.601	0.816	0.930	1.502	0.190
		4	2.777	5.895	0.766	0.852	1.287	0.152
		5	2.510	5.523	0.747	0.809	1.203	0.137
		20	2.032	4.579	0.691	0.698	1.053	0.107
	ITPNet+HiVT	2	2.631	5.703	0.757	0.819	1.218	0.141
	LaneGCN (Liang et al., 2020)	3	3.512	7.607	0.837	1.007	1.642	0.234
		4	3.093	6.805	0.817	0.941	1.520	0.202
		5	2.817	6.401	0.804	0.878	1.417	0.171
		20	2.248	5.209	0.746	0.788	1.191	0.129
	ITPNet+LaneGCN	2	2.922	5.627	0.765	0.894	1.425	0.173
nuScenes	HiVT (Zhou et al., 2022)	2	6.564	13.745	0.914	1.772	2.836	0.505
		3	5.182	11.887	0.908	1.455	2.564	0.445
		4	5.159	11.836	0.903	1.442	2.567	0.442
		5	5.002	11.520	0.899	1.431	2.559	0.419
	ITPNet+HiVT	2	5.514	12.584	0.909	1.503	2.628	0.483
	LaneGCN (Liang et al., 2020)	2	6.125	14.300	0.935	1.878	3.497	0.630
		3	5.853	13.845	0.941	1.697	3.160	0.592
		4	5.708	13.492	0.933	1.653	3.060	0.569
		5	5.663	13.427	0.934	1.647	3.052	0.573
	ITPNet+LaneGCN	2	5.739	13.555	0.919	1.679	3.146	0.580

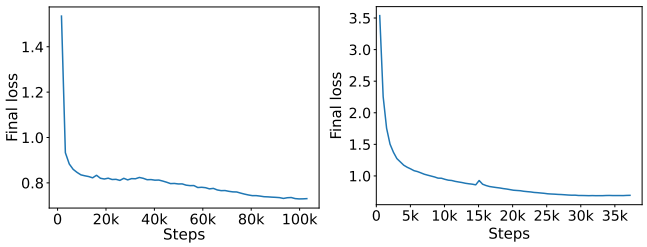


Figure 5: Convergence analysis of our method. Left for Argoverse and right for nuScenes.

A.5 Details about $\mathcal{L}_{reg}$ and $\mathcal{L}_{cls}$ of backbones

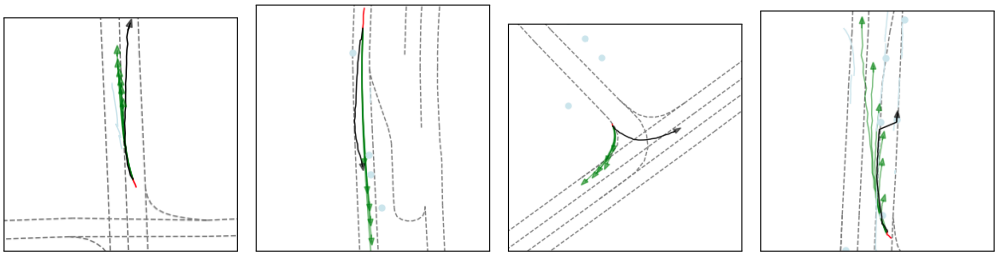
HiVT parameterizes the distribution of future trajectories as a mixture model where each mixture component is a Laplace distribution. The regression loss $\mathcal{L}_{reg}$ is defined as:

$$\mathcal{L}_{reg} = \sum_{i=3}^{M+2} \log \frac{1}{2b} \exp(-\frac{|\hat{x}_i - \mu_i|}{b}), \tag{13}$$

where b is a learnable scale parameter of Laplace distribution, $\hat{x}^i$ is the predicted future trajectory closest to the ground-truth future trajectory and μ_i is the ground-truth future trajectory. The $\mathcal{L}_{cls}$ is defined as cross-entropy loss to optimize the mixing coefficients,

$$\mathcal{L}_{cls} = -\sum_{k=1}^{K} \pi^k \log \hat{\pi}^k, \tag{14}$$

where π^k and $\hat{\pi}^k$ are the probability of the k^{th} trajectory to be selected, and $\pi^k = 1$ if and only if $\hat{\mathbf{X}}^k$ is the predicted future trajectory closest to the ground-truth future trajectory.



(a) (b) (c) (d)

Figure 6: Failure case of ITPNet+HiVT on Argoverse. The observed trajectories are shown in red, the ground-truth trajectories are shown in black, and the predicted multi-modal trajectories are shown in green.

LaneGCN employ smooth L_1 loss as $\mathcal{L}_{reg}$, which is defined as,

$$\mathcal{L}_{reg} = \sum_{i=3}^{M+2} \delta(\hat{x}_i - x_i), \tag{15}$$

where the definition of δ is same as Equation (6). The LaneGCN employs max-margin loss as $\mathcal{L}_{cls}$, which is defined as,

$$\mathcal{L}_{cls} = \frac{1}{K-1} \sum_{k \neq k'} \max(0, \pi^k + \epsilon - \pi^{k'}), \tag{16}$$

where the k^{th} predicted future trajectory is the closest one to the ground-truth future trajectory. This max-margin loss pushes the closest one away from others at least by a margin ϵ.

A.6 Implmentations of Baselines

MOE. To have a fair comparison, we extend the HiVT backbones used in this paper to MOE. We use the A-A Interaction and A-L Interaction in HiVT to as the In-patch Aggregation in MOE and replace the Global interaction in HiVT to replace the Cross-patch Aggregation in MOE. In addition, followed by MOE, we employ the soft-pretraining with masked trajectory complement and Context restoration tasks.

Distill. We also utilize HiVT as the backbone of Distill for fair comparison. In addition, followed by Distill we distill knowledge from the output of the encoder (output of Global Interaction in HiVT) and decoder (output of last hidden layer).