Supplementary Material
JFP: Joint Future Prediction with Interactive Multi-Agent Modeling for Autonomous Driving

1 Demo Video

Please refer to the accompanying video for better and extensive visualization.

2 Ablation Results

Heuristics While we present a learning-based method to learn the pairwise potential for joint future prediction, we can also use simple heuristics to manually construct the pairwise potential. Specifically, for each agent pair, their pairwise potential is set to 1e9 if the corresponding predicted trajectory pair overlaps, otherwise it is set to 0 (thanks to anonymous reviewer for the suggestion). In Tab.1 (as consistent with Tab. 2 in main paper), *Heuristic w/o training* represents applying the heuristic on top of a pretrained model as a post-processing step without training, and *Heuristic w/ training* is retraining the full model using heuristic pairwise potential. We can see that both heuristic methods perform worse than our proposed method.

	Overlap		Marginal				Pairwise		
	All	AV	minADE	minFDE	MissRate	mAP	minADE	minFDE	MissRate
MultiPath++ [32]	0.99	0.05	**0.30**	**0.74**	**0.03**	0.47	0.37	1.00	0.11
JFP [Dynamic]	**0.79**	**0.02**	0.31	0.76	0.04	**0.49**	**0.36**	0.96	0.11
Heuristic w/o training	0.75	0.02	0.31	0.76	0.04	0.49	0.45	1.23	0.16
Heuristic w/ training	1.16	0.06	0.52	1.46	0.11	0.42	0.87	2.20	0.19

Table 1: Ablation about using heuristic pairwise potentials on WOMD extended setting.

Wayformer Backbone We experimented with the latest SOTA backbone of wayformer [33] on the WOMD interaction validation dataset (as consistent with Tab. 1 in the main paper). As shown in Tab. 2, the improvement of JFP using Wayformer backbone over the bare Wayformer backbone (last two rows) is also clear across all joint metrics.

Method	Overlap(↓)	minSADE(↓)	minSFDE(↓)	SMissRate(↓)	mAP(↑)
SceneTransformer(M) [19]	0.091	1.12	2.60	0.54	0.09
SceneTransformer(J) [19]	0.046	0.97	2.17	0.49	0.12
MultiPath++ [32]	0.064	1.00	2.33	0.54	0.18
JFP [AV-Star] (MultiPath++ backbone)	0.030	**0.87**	**1.96**	**0.42**	**0.20**
Wayformer [33]	0.061	0.99	2.30	0.47	0.16
JFP [AV-Star] (Wayformer backbone)	**0.017**	0.95	2.17	0.45	0.17

Table 2: Ablation using Wayformer [33] backbone on WOMD interactive validation dataset.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

3 Gradient Derivation

In this section, we complete the derivation of gradient computation in Sec. 3.4 for

$$\mathcal{L}_{E_{traj}} = \text{cross\_entropy}(\mu_i + \text{stop\_gradient}(\hat{\mu}_i - \mu_i))$$
$$\mathcal{L}_{E_{pair}} = \text{cross\_entropy}(\upsilon_{i,j} + \text{stop\_gradient}(\hat{\upsilon}_{i,j} - \upsilon_{i,j}))$$

as part of gradient computation for Eq. 2. Here we omit $\mathcal{X}, \theta$ for simplicity and denote μ_i as the unary softmax logits, υ_{ij} as the pairwise softmax logits, *i.e.* $\mu_i = \log(q(\mathbf{s}_i|\mathcal{X}, \theta)) = -E_{traj}(\mathbf{s}_i|\mathcal{X}, \theta))$, $\upsilon_{ij} = -E_{pair}(\mathbf{s}_i, \mathbf{s}_j|\mathcal{X}, \theta)$, and $\hat{\mu}_i, \hat{\upsilon}_{i,j}$ as the corresponding marginal logits after message passing. Following a similar procedure in Training CRFs, we can compute the derivative of $-\log p(\mathbf{s})$ over $\mu_i(k)$, where $\mu_i(k)$ is i^{th} agent's k^{th} state:

$$\begin{aligned}
\frac{\partial - \log p(\mathbf{s})}{\partial \mu_i(k)} &= -\frac{\partial}{\partial \mu_i(k)} \left(\sum_i \mu_i(\mathbf{s}) + \sum_{(i,j)\in\mathcal{G}} \upsilon_{i,j}(\mathbf{s}) - \log \mathcal{Z} \right) \\
&= -\mathbb{1}_{\mathbf{s}_i=k} + \frac{\partial \log \mathcal{Z}}{\partial \mu_i(k)} \\
&= -\mathbb{1}_{\mathbf{s}_i=k} + \frac{1}{\mathcal{Z}} \cdot \frac{\partial}{\partial \mu_i(k)} \sum_{\hat{\mathbf{s}}:\hat{\mathbf{s}}_i=k} \exp(-E(\hat{\mathbf{s}})) \\
&= -\mathbb{1}_{\mathbf{s}_i=k} + \sum_{\hat{\mathbf{s}}:\hat{\mathbf{s}}_i=k} \frac{1}{\mathcal{Z}} \cdot \frac{\partial}{\partial \mu_i(k)} \exp(-E(\hat{\mathbf{s}})) \\
&= -\mathbb{1}_{\mathbf{s}_i=k} + \sum_{\hat{\mathbf{s}}:\hat{\mathbf{s}}_i=k} \frac{\exp(-E(\hat{\mathbf{s}}))}{\mathcal{Z}} \cdot \frac{\partial}{\partial \mu_i(\hat{\mathbf{s}})} \left(\sum_i \mu_i(\hat{\mathbf{s}}) + \sum_{(i,j)\in\mathcal{G}} \upsilon_{i,j}(\hat{\mathbf{s}}) \right) \\
&= -\mathbb{1}_{\mathbf{s}_i=k} + \sum_{\hat{\mathbf{s}}:\hat{\mathbf{s}}_i=k} p(\hat{\mathbf{s}})
\end{aligned}$$

Note $\sum_{\hat{\mathbf{s}}:\hat{\mathbf{s}}_i=k} p(\hat{\mathbf{s}})$ is the marginal outputs $\hat{\mu}_i$ for state k. We can recognize that the final gradient takes the form exactly as a cross entropy loss on $\hat{\mu}_i$. Thus optimizing the objective function in Eq. 2. w.r.t. μ_i is equivalent as applying cross entropy loss $\mathcal{L}_{E_{traj}}$ on $\hat{\mu}_i$: $\mathcal{L}_{E_{traj}} = \text{cross\_entropy}(\mu_i + \text{stop\_gradient}(\hat{\mu}_i - \mu_i))$. Note the stop-gradient is used to ensure that the computation is from $\hat{\mu}_i$ but the gradients are on μ_i and can continue flow into the backbone network. Similar derivation applies to $\upsilon_{i,j}$ for using loss $\mathcal{L}_{E_{pair}}$.