Foresight in Motion: Reinforcing Trajectory Prediction with Reward Heuristics

Supplementary Material

6. Implementation Details

This section provides additional details, including query-centric inverse reinforcement learning (QIRL), Bi-Mamba-enhanced decoder, loss function, and training specifics.

6.1. Query-Centric IRL

In our reward-driven intention reasoner, the IRL problem is modeled as a Markov Decision Process (MDP). Given a demonstrated trajectory (or path, if available), we discretize it into a sequence of states on a 2D grid. Our QIRL framework integrates Maximum Entropy IRL (MaxEnt IRL) with a query-centric paradigm. Specifically, each grid cell is treated as a state s_i, corresponding to the grid query $Q_G^{s_i}$ at the position $s_i = (x_i, y_i)$. The action a represents movement to an adjacent grid cell, and the process is assumed to be deterministic.

Let $\mathcal{D} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_N\}$ denote a set of demonstrations, where each traversal (or *plan*) ς_i consists of a sequence of states $\varsigma_i = \{s_1, s_2, \ldots, s_{\mathcal{H}}\}$, with $\mathcal{H}$ representing the planning horizon. The objective of IRL is to infer the reward function R that explains these demonstrations [5]. MaxEnt IRL [9] addresses this problem using the maximum entropy principle, modeling the probability of a *plan* as:

$$P(\varsigma|R) \propto \exp\left(\sum_{s_i, a_i \in \varsigma} R(s_i, a_i)\right). \tag{1}$$

Consequently, solving the IRL problem can be formulated as a Maximum-A-Posteriori (MAP) estimation, aiming to maximize the joint posterior distribution of the expert demonstrations. The log-likelihood is expressed as:

$$\begin{aligned} L(\theta) &= \log\left(P(\mathcal{D}, \theta|R)\right) \\ &= \log\left(P(\mathcal{D}|R)\right) + \log\left(P(\theta)\right), \end{aligned} \tag{2}$$

where θ represents the parameters of the reward neural network. By ignoring the regularization term $\log(P(\theta))$, we derive the gradient of the log-likelihood as:

$$\frac{\partial L(\theta)}{\partial \theta} = \frac{\partial \log\left(P(\mathcal{D}|R)\right)}{\partial \theta} = \frac{\partial \log\left(P(\mathcal{D}|R)\right)}{\partial R} \cdot \frac{\partial R}{\partial \theta}. \tag{3}$$

The second term corresponds to the back-propagation of the reward network, while the first term, as derived in [6], is calculated as the difference between the state visitation counts from the expert demonstrations and the expected visitation counts induced by the learned reward distribution.

Once the reward function R is obtained, it provides the action preferences for each grid cell. Leveraging this reward distribution, we can generate multiple plausible intention sequences by sampling rollouts, following the approach in [7]. These intention sequences serve as reasoning priors for subsequent trajectory prediction.

Algorithm 1: Bi-Mamba Decoder Process

Input: Trajectory token Q_T,
Dual mode token $Q_M = \{\text{CLS}_1, \text{CLS}_2\}$.
Output: Trajectory offset ΔY, and probability p.

$\text{T} = \texttt{Norm}([\text{CLS}_1; Q_T; \text{CLS}_2])$
$\text{x} \leftarrow \texttt{Linear}^{\text{x}}(\text{T})$
$\text{z} \leftarrow \texttt{Linear}^{\text{z}}(\text{T})$
for o in {Forward, Backward} **do**
  $\text{x}'_o \leftarrow \texttt{SiLU}(\texttt{Conv1d}_o(\text{x}))$
  $\text{B}_o \leftarrow \texttt{Linear}_o^{\text{B}}(\text{x}'_o)$
  $\text{C}_o \leftarrow \texttt{Linear}_o^{\text{C}}(\text{x}'_o)$
  $\Delta_o \leftarrow \log(1 + \exp(\texttt{Linear}_o^{\Delta}(\text{x}'_o) + \texttt{Param}_o^{\Delta}))$
  $\overline{\text{A}}_o \leftarrow \Delta_o \otimes \texttt{Param}_o^{\text{A}}$
  $\overline{\text{B}}_o \leftarrow \Delta_o \otimes \text{B}_o$
  $\text{h}_o \leftarrow 0, \text{y}_o \leftarrow 0$
  for i in {0, ... M-1} **do**
    $\text{h}_o = \overline{\text{A}}_o[:, i, :, :] \odot h_o +$
      $\overline{\text{B}}_o[:, i, :, :] \odot \text{x}'_o[:, i, :, \text{None}]$
    $\text{y}_o[:, i, :] = h_o \otimes \text{C}_o[:, i, :]$
  end
end
$\text{y}'_{\text{Forward}} \leftarrow \text{y}_{\text{Forward}} \odot \texttt{SiLU}(\text{z})$
$\text{y}'_{\text{Backward}} \leftarrow \text{y}_{\text{Backward}} \odot \texttt{SiLU}(\text{z})$
$\text{T}' \leftarrow \texttt{Linear}^{\text{T}}(\text{y}'_{\text{Forward}} + \text{y}'_{\text{Backward}}) + \text{T}$
$[\text{CLS}'_1; Q'_T; \text{CLS}'_2] \leftarrow \texttt{Norm}(\text{T}')$
$\text{CLS}' \leftarrow \texttt{SelfAttn}(\text{CLS}'_1 + \text{CLS}'_2)$
$p \leftarrow \texttt{SoftMax}(\texttt{Linear}^{Q_M}(\text{CLS}'))$
$\Delta Y \leftarrow \texttt{Linear}^{Q_T}(Q'_T)$

6.2. Bi-Mamba-Enhanced Decoder

We detail the Bi-Mamba decoding process. Given the trajectory token $Q_T \in \mathbb{R}^{K \times T_f \times C}$ and the dual mode token $Q_M = \{\text{CLS}_1, \text{CLS}_2\} \in \mathbb{R}^{K \times 2 \times C}$, we first concatenate these inputs, followed by a normalization layer, to form a sequence token $\text{T} \in \mathbb{R}^{K \times (T_f+2) \times C}$. This sequence token is then passed into the bidirectional Mamba model [8].

Specifically, in the Mamba architecture [1], the structured state-space model (SSM) is represented as a continuous system defined by ($\Delta \in \mathbb{R}^{K \times T_f \times C}, \text{A} \in \mathbb{R}^{C \times D}, \text{B} \in \mathbb{R}^{K \times T_f \times D}, \text{C} \in \mathbb{R}^{K \times T_f \times D}$), where D represents the dimension of the hidden state. To enable further processing, this continuous system needs to be discretized into its counterpart ($\overline{\text{A}} \in \mathbb{R}^{K \times T_f \times C \times D}, \overline{\text{B}} \in \mathbb{R}^{K \times T_f \times C \times D}, \text{C}$).

We begin by applying a linear projection to the normalized sequence token, yielding the state vector x and the gate vector z. The state vector x is then processed in both forward and backward directions. For each direction, x is encoded via a 1-D CNN and subsequently projected into $(\mathrm{B}_o, \mathrm{C}_o, \Delta_o)$. Through a series of transformation operations, the discretized matrices $\overline{\mathrm{A}}_o$ and $\overline{\mathrm{B}}_o$ are generated and utilized in the SSM's recurrent process to compute the bidirectional outputs $\mathrm{y}_{\text{Forward}}$ and $\mathrm{y}_{\text{Backward}}$. These outputs are gated by z, integrated with a residual connection to the original sequence token, and normalized, ultimately producing the updated dual mode tokens and sequential trajectory tokens. Finally, a self-attention layer is applied to fuse the mode features, followed by a linear projection and a softmax operation to compute the probability. Additionally, the trajectory offset is derived through a simple linear projection. The entire decoding process is detailed in Algorithm 1.

6.3. Loss Function

We present the training loss with detailed mathematical formulations, encompassing the Occupancy Grid Map (OGM) loss, regression loss, and classification loss.

Occupancy Grid Map Loss. Let the output of the OGM generator be $\mathcal{O} \in \mathbb{R}^{H \times W \times T_f}$ and the Ground Truth (GT) occupancy grid be $\mathcal{O}^{\text{GT}} \in \mathbb{R}^{H \times W \times T_f}$. We employ a modified focal binary cross-entropy loss [4], expressed as:

$$\mathcal{L}_{OGM} = -\frac{1}{n^*} \sum_{i=1}^{H \times W} \{(o_i)^{\mu}(1 - o_i^{\text{GT}})^{\nu} \log(1 - o_i) + (1 - o_i)^{\mu}(o_i^{\text{GT}})^{\nu} \log(o_i)\}, \tag{4}$$

where $o_i \in \mathcal{O}$ is the predicted occupancy value for each grid cell, and $o_i^{\text{GT}} \in \mathcal{O}^{\text{GT}}$ is the corresponding GT label. A value greater than 0.85 is considered a positive hypothesis. The total number of positive classes is denoted as n^*, and the focusing factors are set to $\mu = 2$ and $\nu = 4$.

Regression Loss. Given the trajectory proposals $\overline{Y} \in \mathbb{R}^{K \times T_f \times 2}$, the final predicted trajectories $Y \in \mathbb{R}^{K \times T_f \times 2}$, and the GT trajectory $Y^{\text{GT}} \in \mathbb{R}^{T_f \times 2}$, the regression loss is calculated using a modified smooth ℓ_1 loss (Huber loss) with a Winner-Takes-All (WTA) strategy. Among the K trajectories, we identify the one with the minimum Euclidean distance to the GT as the positive trajectory, denoted $\overline{Y}^*$ and Y^*, respectively. The regression loss is then computed as:

$$\mathcal{L}_{REG} = \frac{1}{T_f} \left(\|\mathbb{H}(\overline{Y}^* - Y^{\text{GT}})\|_1 + \|\mathbb{H}(Y^* - Y^{\text{GT}})\|_1 \right), \tag{5}$$

where $\mathbb{H}$ applies the Huber function element-wise to δY as:

$$\mathbb{H}(\delta Y_i) = \begin{cases} 0.5\|\delta Y_i\|_2^2 & \text{if } \|\delta Y_i\|_2 < 1, \\ \|\delta Y_i\|_2 - 0.5 & \text{otherwise,} \end{cases} \tag{6}$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ represent the ℓ_1-norm and ℓ_2-norm, respectively.

Classification Loss. We adopt a max-margin loss for classification as proposed in [2]. Let $p \in \mathbb{R}^{K \times 1}$ denote the predicted probabilities for each trajectory. The positive class p^* is determined by identifying the trajectory whose endpoint is closest to the Ground Truth (GT) in terms of Euclidean distance. The classification loss is then formulated as:

$$\mathcal{L}_{CLS} = \frac{1}{K-1} \sum_{p_i \neq p^*} \max(0,\ p_i - p^* + \epsilon), \tag{7}$$

where $p_i \in p,\ i = 1, 2, \ldots, K$ denotes the probability of each trajectory, and ϵ is the margin which is set to 0.2.

6.4. Training Details

For the Argoverse dataset, the number of modes is set to $K = 6$, while for the nuScenes dataset, $K = 10$. All hidden feature dimensions are configured as $C = 128$. The total model size is approximately 5.2M parameters. The model is trained end-to-end on eight GPUs with a total batch size of 128 for 40 epochs. We adopt the AdamW [3] optimizer with a weight decay coefficient of 1×10^{-4}. The learning rate is initialized at 1×10^{-4} and is gradually reduced to 1×10^{-5} after 30 epochs. Note that no data augmentation is applied during the training process.

7. Additional Qualitative Results

This section presents additional qualitative results for challenging cases from the Argoverse validation set, as illustrated in Fig. 1. The first three rows demonstrate that our approach effectively captures multimodality, successfully predicting multiple plausible trajectories while considering the scene layout in various complex scenarios. In the final row, when the motion trend is clear, our model reliably follows the trend to produce reasonable predictions. Notably, even in situations where the agent might enter a parking lot, as shown in the last column, the model can still provide accurate and plausible trajectory predictions.

References

[1] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1

[2] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556, 2020. 2

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 2

[4] T.-Y. Ross and P. Dollár. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988, 2017. 2

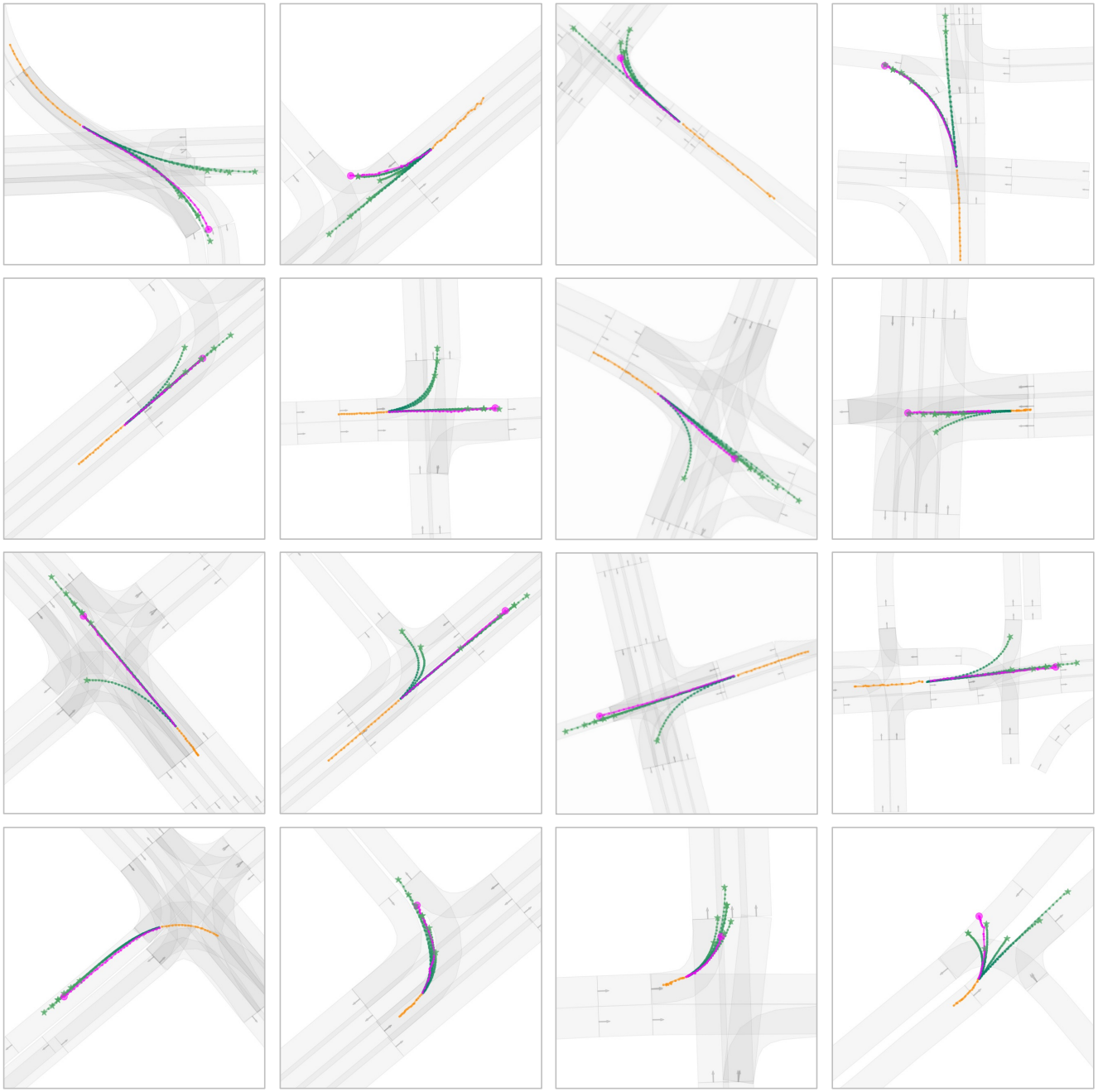

Figure 1. Additional qualitative results of our model on the Argoverse validation set. The historical trajectory, ground-truth future trajectory, and multimodal predictions are depicted in yellow, magenta, and green, respectively. Other traffic participants are omitted to better highlight the predictions for the target agent.

[5] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998. 1

[6] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36 (10):1073–1087, 2017. 1

[7] Yanfu Zhang, Wenshan Wang, Rogerio Bonatti, Daniel Maturana, and Sebastian Scherer. Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories. In *Conference on Robot Learning*, pages 894–905, 2018. 1

[8] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space

model. *arXiv preprint arXiv:2401.09417*, 2024. 1

[9] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438. Chicago, IL, USA, 2008. 1