T4P: Test-Time Training of Trajectory Prediction via Masked Autoencoder and Actor-specific Token Memory

Supplementary Material

1. Metric definitions

minimum Average Displacement Error (mADE) The ADE measures the average L2 distance between the predicted trajectory $\hat{\mathbf{x}}_t^n = (x^n_{t-t_h:t+t_f}, y^n_{t-t_h:t+t_f})$ and its corresponding ground truth $\mathbf{x}_t^n$ for n-th agent and t-th time step. The mADE$_k$ represents the minimum ADE over the k most likely predictions, and is found for all scenes **S** in the test set.

$$\text{ADE} = \frac{1}{|\mathbf{S}|}\sum_{s=0}^{|\mathbf{S}|}\frac{1}{N}\sum_{n=0}^{N}\frac{1}{T_s}\sum_{t=0}^{T_s}\|\mathbf{x}_t^n - \hat{\mathbf{x}}_t^n\|_2 \tag{1}$$

$$\text{mADE}_k = \min_k(\text{ADE}_{(1)}, ..., \text{ADE}_{(k)}) \tag{2}$$

minimum Final Displacement Error (mFDE) The FDE measures the L2 distances between the predicted final point $\hat{\text{x}}^n_{T_s} = (x^n_{T_s}, y^n_{T_s})$ of the prediction and ground truth. The mFDE$_k$ represents the minimum FDE over the k most likely predictions, and is found for all scenes **S** in the test set.

$$\text{FDE} = \frac{1}{|\mathbf{S}|}\sum_{s=0}^{|\mathbf{S}|}\frac{1}{N}\sum_{n=0}^{N}\|\text{x}^n_{T_s} - \hat{\text{x}}^n_{T_s}\|_2 \tag{3}$$

$$\text{mFDE}_k = \min_k(\text{FDE}_{(1)}, ..., \text{FDE}_{(k)}) \tag{4}$$

Miss Rate (MR) MR is the proportion of missed predictions over all predictions. Following the nuScenes dataset, we defined the prediction whose maximum pointwise L2 distance to ground truth is greater than 2 meters as missed predictions. MR$_k$ take the k most likely predictions and determine whether they are missed predictions or not. If there are m misses over total n predictions, MR would be $\frac{m}{n}$.

2. Model details

We employ the same model architecture as our backbone model, ForecastMAE [2]. Actor-specific tokens, represented by $\bar{\alpha} \in \mathbb{R}^{C\times D}$, comprise learnable parameters for each class (C).

$$\bar{\alpha} \in \mathbb{R}^{C\times D} : \left\{\alpha(c) \in \mathbb{R}^D\right\}^C \tag{5}$$

Each n^{th} actor corresponds to an actor class token $\alpha_n(c)$ within a specific class, where during offline training, all actors within the same class share the same class token $\alpha(c)$ regardless of individual instances. During test-time training and online evaluation, leveraging historical motion patterns for each actor instance at a specific time (t), we refine the actor-specific token $\alpha_n^t(c)$ to capture distinct actor-specific motion using Algorithm 1.

In our MAE training approach, we implement random masking for lane and complementary masking for actors. Random lane masking involves replacing a segment of lane embeddings with a learnable lane masking token. Complementary actor masking entails randomly selecting a subset of actors, replacing their future trajectory embeddings with a learnable trajectory masking token. Other actors have their past trajectory embeddings replaced with the same learnable trajectory masking token.

3. Training details

As our focus is set on test-time training for trajectory prediction, baseline comparisons are made with existing *test-time adaptation methods*.

DUA. Retaining the hyperparameters from the official implementation, we use *pre_momentum* (ρ_0) = 0.1, *decay_factor* (w) = 0.94, and *min_momentum_constant* (ζ) = 0.005 for momentum updates.

TENT w/ sup. Employing identical hyperparameters, including learning rate and weight decays, we update layers of BatchNorm1d, BatchNorm2d, BatchNorm3d, SyncBatchNorm, and LayerNorm types within the backbone model.

MEK. Adhering to the online learning methodology from the original paper [7] and importing the MEK optimizer from *MEKF-MAME* [1]. In the absence of multi-modal prediction loss mention, we utilize the WTA loss [3], commonly employed for multi-modal trajectory prediction.

AML. Employing the hyperparameters and methodology from the official implementation for the nuS $\rightarrow$ Lyft experiment, adapted to our backbone. We set α_{init} = 0.0001, learning_rate = 0.001, and sigma_eps$_{init}$ = 0.102. The regression loss of AML is different from ours and MEK. While ours and MEK use delayed historical and future trajectories ($\mathcal{X}_{t-\tau}, \mathcal{Y}_{t-\tau}$), AML uses current historical trajectory ($\mathcal{X}_t$) for compute regression loss for meta learning. It has advantage in using current motion, but disadvantage in not using full historical and future trajectories.

Ours We jointly train regression and reconstruction losses with equal weightage (set as 1). The reconstruction loss encompasses history, future, and lane reconstructions with weights of 1, 1, and 0.35, respectively. We update all types of layers across all depths.

4. Algorithm of actor-specific token memory

The actor-specific token is implemented as a learnable embedding of a transformer, and stored in a memory dictionary where actor instance ID/corresponding tokens are key/values; as each token is a 128-dim vector, countless actors can be stored in memory.

While we know when an actor disappears and reappears during offline training, this is unavailable during online training/inference. How the token memory evolves is closely related to how an object-tracking network tracks actor IDs during online training and inference. During online training/inference, actor IDs are tracked by an object-tracking network. In the case where an actor disappears/reappears and the tracker succeeds in restoring the actor ID, the ID is used to retrieve the corresponding actor-specific token from the dictionary. However, when the tracker fails and assigns a new ID, our method also re-initializes the token. If an actor appears, an object tracker has its own strategy for actor-instance initialization. As the proposed protocol can share the strategy of the tracker, our method can be integrated into the perception system seamlessly.

The comprehensive algorithm delineating the actor-specific memory is outlined in Algorithm 1. This specialized memory repository encompasses individual actor-specific tokens, dynamically evolving as scene-relative time passes. Upon a scene transition, the actor-specific tokens are collected by class, averaged, and subsequently passed on to the succeeding scene.

Algorithm 1 Actor-Specific Token Memory

```
Given $s$ = the scene index of a set of scenes $\mathbf{S}$,
$n$ = the $n^{th}$ actor seen in the scene,
$c$ = the class index from a set of $C$ classes,
$\alpha$ = actor specific tokens,
$t$ = scene relative time,
$T_s$ = time length for scene $s$,
$(c)$ = the class type of actor,
$N_c$ = the number of actors of class $c$
$E$ = the network Encoder
$D$ = the network Decode, and
$\tau$ = the delayed time stamp:
 1: $\bar{\alpha}_{\textbf{scene(0)}}(c) \leftarrow \bar{\alpha}_{\textbf{train}}(c),\ \forall c \in C$
 2: for $s = 0 : |\mathbf{S}|$ do
 3:     $n \leftarrow 0$
 4:     $\boldsymbol{\mathcal{A}} \leftarrow \{\}$                ▷ This serves as the memory bank
 5:     for $t = 0 : T_s$ do
 6:         for $\alpha_{new}$ do            ▷ $\forall$ new actor $\alpha_{new}$ at time t
 7:             $\alpha_n^t(c) \leftarrow \bar{\alpha}_{\textbf{scene}(s)}(c)$
 8:             $\boldsymbol{\mathcal{A}}.insert(\alpha_n^t(c))$
 9:             $n \leftarrow n + 1$
10:         end for
11:         if $t \bmod \tau \equiv 0$ then          ▷ Train every $\tau$ steps
12:             if $\alpha_n^t(c) \in \textbf{scene}(s)_t$ then
13:                 $\alpha_n^{t+1}(c) \leftarrow \alpha_n^t(c)$-$\frac{\partial \mathbb{E}^{t-\tau}(\alpha_n(c), \mathcal{X}, \mathcal{Y}, \mathcal{M})}{\partial \alpha_n^{t-\tau}}$
14:                             $\forall\ \alpha_n^t(c) \in \boldsymbol{\mathcal{A}}$
15:             else
16:                 $\alpha_n^{t+1}(c) \leftarrow \alpha_n^t(c)$
17:                             $\forall\ \alpha_n^t(c) \in \boldsymbol{\mathcal{A}}$
18:             end if
19:         end if
20:         $\mathcal{Y}_t = D(E(\mathcal{X}_t, \mathcal{M}_t, \boldsymbol{\mathcal{A}}))$              ▷ Online-Eval
21:     end for
22:     for $c = 0 : |C|$ do
23:         $\bar{\alpha}_{\textbf{scene(s+1)}}(c) \leftarrow \frac{1}{N_c} \sum_n^{N_c} \alpha_n^{T_s}(c)$
24:     end for
25: end for
```

5. Datasets

We construct four different datasets in the same format using *trajdata* [4], a unified framework for trajectory prediction data. Our preprocessing method aligns closely with the data preprocessing approach of *Forecast-MAE* [2], with a couple of modifications from the original methodology.

Primarily, we omit certain information used in the original method, such as *is_intersections* and *lane_attribute*, due to their non-universal availability across all datasets. Additionally, our lane parsing method diverges in specifics. We focus on lane information within a 50-meter radius of ego-agents, interpolating each lane centerline to standardize point distances within the lane to 1 meter. Furthermore, individual lanes are divided into segments, each segment limited to a maximum length of 20 meters.

The parsing of data within a scenario, facilitated by the scenario-based parsing protocol of *trajdata*, hinges on parsing time configuration and absolute scenario length. Our method utilizes two distinct time configurations for prediction purposes: a 0.9-second past (including the current second) and a 3.0-second future interval with a 0.1-second time interval for short-term prediction. Conversely, for long-term prediction, we opt for a 2.0-second past (including the current second) and a 6.0-second future interval with a 0.5-second time interval. This configuration results in shorter time intervals yielding longer scenario lengths for short-term prediction compared to longer time intervals for long-term prediction. Considering sequential nature of driving scenario, we use first 10,000 samples for the evaluation.

Tables 1 and 2 display the mean scenario lengths across datasets for long-term and short-term prediction, respectively. Our update step (τ) is set as t_f, ensuring that the target data encompasses a scenario length exceeding t_f (12 for long-term and 30 for short-term). Consequently, for long-term prediction, nuS and Lyft datasets are designated as target datasets, while for short-term prediction, nuS, Lyft, and Way datasets serve as the target datasets. Please note that the real-world application is a continuous setting without scene transition, so setting the target dataset as a sufficiently long scenario length is a realistic experiment setting.

Table 1. Mean scenario length of each *val* dataset in long-term prediction configurtaion.

Long-term (2/6/0.5)	nuS	Lyft	Way
scene length	23.0	32.7	2.0

Table 2. Mean scenario length of each *val* dataset in short-term prediction configurtaion.

Short-term (1/3/0.1)	nuS	Lyft	Way	INTER
scene length	150.4	200.0	50.5	6.3

6. Further quantitative results

6.1. Additional cross-dataset adaptation results

We conducted exhaustive cross-dataset adaptation experiments for both long-term and short-term predictions, detailed in Tab.1 and Tab.2, respectively. These results notably demonstrate the distinct superiority of our method over existing Test-Time Adaptation (TTA) and online learning methodologies.

6.2. More metrics comparison

mADE_1 and Missrate results in Tab. 3 show ours is still effective. Missrate (MR) is widely-used and measures precision.

Table 3. Comparison with other metrics (lower is better)

(mADE_1/Missrate)	Source only	TENT	MEK	Ours
INTER $\rightarrow$ nuS (1/3/0.1)	3.030 / 0.338	1.726 / 0.336	1.461 / 0.291	**1.239/ 0.154**
nuS $\rightarrow$ Lyft (2/6/0.5)	2.546 / 0.362	2.473 / 0.346	2.412 / 0.323	**1.918 / 0.227**

6.3. More ablation of actor-specific token on other datasets

We included only two datasets due to page limits; experiments on other datasets (Lyft, Way) in Tab. 4 still show that our method is effective.

Table 4. Effect of actor-specific memory on additional datasets

Exp (mADE_6/mFDE_6)	Baseline	Ours w/o A-s	Ours
Lyft $\rightarrow$ nuS (1/3/0.1)	0.506 / 1.102	0.420 / 0.934	**0.357 / 0.770**
Way $\rightarrow$ Lyft (2/6/0.5)	0.638 / 1.404	0.594 / 1.311	**0.549 / 1.171**

6.4. Efficiency under long-term configuration

In Fig. 1, we present another efficiency comparison experiment between our method and baseline approaches in the long-term configuration. Demonstrating superiority in both accuracy and efficiency, our method outperforms the baseline methods in this setting. All methods achieve real-time execution, surpassing the 2FPS benchmark set by the 0.5 time interval of data acquisition. However, in the long-term configuration with an adaptation step of 5, MEK shows a 17.7 improvement in computational efficiency but experiences a significant decline in accuracy. TENT's efficiency remains nearly consistent with the short-term configuration, but its accuracy doesn't exhibit significant improvement from joint training, making it noteworthy in this context.

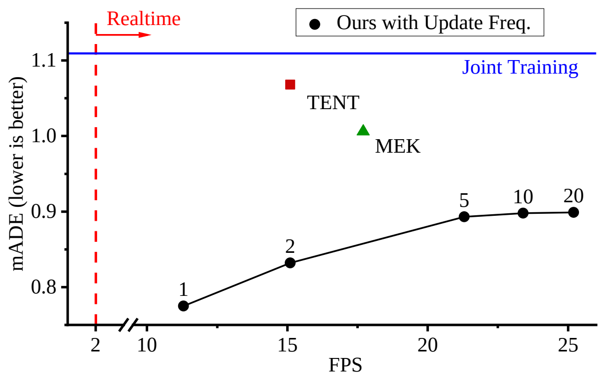

Figure 1. mADE_6 and FPS of our method and the baselines in nuS $\rightarrow$ Lyft long-term experiment (2/6/0.5).

7. Expanded visual results

7.1. Comparative analysis with baseline methods

Figure. 2 showcases additional qualitative comparison results between our method and the baseline methods. Notably, TENT w/ sup exhibits minimal adaptation in comparison to the other methodologies. Regarding MEK, the supervision signal tends to lead to underfitting or over-adaptation, primarily adjusting the last layer weights of the decoder without inducing the model to acquire robust representations. Conversely, our method demonstrates stable adaptability, particularly excelling in challenging scenarios.

7.2. Reconstruction results

In Fig. 3, additional reconstruction results are showcased. These reconstruction examples are trained offline using the source dataset and subsequently adapted during test time using the target dataset via reconstruction loss. Upon examination of the three reconstruction instances, it's evident that the method appropriately learns lane structures, emphasizing an understanding of interactions among lane segments and actor trajectories. Furthermore, in the second column, the proper reconstruction of future trajectories for actors is observed, showcasing an understanding of both lane structure and the motion of following/leading actors.

7.3. Extended multi-modal prediction findings

Considering importance of multi-modal prediction [5, 6], figure. 4 presents additional multi-modal prediction results obtained through our adaptation method. In the first and second columns, our approach generates diverse and plausible prediction samples while comprehending road structures adeptly. Notably, it showcases an understanding of complex road structures, such as turn scenarios. Moving to the third column, our method also demonstrates an understanding of interactions with surrounding actors. It illustrates instances where our model generates trajectories that avoid collisions by either surpassing nearby actors or transitioning to another lane. In contrast, the non-updated version produces trajectories that could lead to collisions with the front right actor.

References

[1] Abulikemu Abuduweili and Changliu Liu. Robust online model adaptation by extended kalman filter with exponential moving average and dynamic multi-epoch strategy. In *Learning for Dynamics and Control*, pages 65–74. PMLR, 2020. 1

[2] Jie Cheng, Xiaodong Mei, and Ming Liu. Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8679–8689, 2023. 1, 2

[3] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. *Advances in neural information processing systems*, 25, 2012. 1

[4] Boris Ivanovic, Guanyu Song, Igor Gilitschenski, and Marco Pavone. trajdata: A unified interface to multiple human trajectory datasets. In *Proceedings of the Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, New Orleans, USA, 2023. 2

[5] Daehee Park, Jaewoo Jeong, and Kuk-Jin Yoon. Improving transferability for cross-domain trajectory prediction via neural stochastic differential equation. *arXiv preprint arXiv:2312.15906*, 2023. 4

[6] Daehee Park, Hobin Ryu, Yunseo Yang, Jegyeong Cho, Jiwon Kim, and Kuk-Jin Yoon. Leveraging future relationship reasoning for vehicle trajectory prediction. In *The Eleventh International Conference on Learning Representations*, 2023. 4

[7] Letian Wang, Yeping Hu, and Changliu Liu. Online adaptation of neural network models by modified extended kalman filter for customizable and transferable driving behavior prediction, 2022. 1

Table 5. All cross-dataset adaptation experiments in long-term configuration. We use all nuS, Lyft, Way for the source dataset, and use nuS, Lyft for the target datasets, ensuring they exceed the adaptation step of 12.

mADE_6	Long-term exp (2/6/0.5)				
/ mFDE_6	nuS $\rightarrow$ Lyft	Way $\rightarrow$ Lyft	Way $\rightarrow$ nuS	Lyft $\rightarrow$ nuS	Mean
Source Only	1.122 / 2.577	0.621 / 1.347	1.153 / 2.220	1.434 / 3.178	1.083 / 2.331
Joint Training	1.108 / 2.597	0.638 / 1.404	1.091 / 2.031	1.398 / 3.109	1.059 / 2.285
DUA	1.365 / 3.257	0.790 / 1.868	1.270 / 2.634	1.585 / 3.607	1.253 / 2.842
TENT (w/ sup)	1.068 / 2.514	0.628 / 1.381	1.077 / 2.012	1.395 / 3.102	1.042 / 2.252
MEK ($\tau = t_f/2$)	1.079 / 2.597	0.629 / 1.404	1.079 / 2.031	1.396 / 3.109	1.046 / 2.285
MEK ($\tau = t_f$)	1.006 / 2.369	0.615 / 1.351	1.117 / 2.140	1.426 / 3.119	1.041 / 2.245
AML (K_0)	1.787 / 3.067	1.322 / 2.571	1.618 / 2.999	1.866 / 3.494	1.648 / 3.033
AML (*full*)	1.462 / 2.573	0.977 / 2.184	1.495 / 2.978	1.698 / 3.367	1.408 / 2.776
Ours	**0.776 / 1.820**	**0.549 / 1.171**	**0.996 / 1.784**	**1.254 / 2.802**	**0.891 / 1.888**

Table 6. All cross-dataset adaptation experiments in shoft-term configuration. We use all nuS, Lyft, Way, INTER for the source dataset, and use nuS, Lyft, Way for the target datasets, ensuring they exceed the adaptation step of 30.

mADE_6	Short-term exp (1/3/0.1)									
/ mFDE_6	INTER $\rightarrow$ nuS	INTER $\rightarrow$ Lyft	INTER $\rightarrow$ Way	nuS $\rightarrow$ Lyft	nuS $\rightarrow$ Way	Way $\rightarrow$ Lyft	Way $\rightarrow$ nuS	Lyft $\rightarrow$ nuS	Lyft $\rightarrow$ Way	Mean
Source Only	1.047 / 2.247	1.391 / 2.945	0.458 / 1.052	0.233 / 0.553	0.431 / 1.031	0.191 / 0.424	0.382 / 0.761	0.484 / 1.096	0.621 / 1.348	0.582 / 1.273
Joint Training	1.116 / 2.445	1.553 / 3.458	0.398 / 1.028	0.233 / 0.504	0.472 / 1.125	0.185 / 0.415	0.374 / 0.718	0.506 / 1.102	0.918 / 1.935	0.639 / 1.414
DUA	1.118 / 2.455	1.516 / 3.352	0.398 / 1.031	0.287 / 0.724	0.516 / 1.294	0.197 / 0.433	0.396 / 0.781	0.474 / 1.118	0.981 / 2.081	0.654 / 1.474
TENT (w/ sup)	1.102 / 2.423	1.519 / 3.405	0.375 / 0.988	0.226 / 0.485	0.448 / 1.071	0.176 / 0.398	0.358 / 0.695	0.462 / 1.009	0.894 / 1.903	0.618 / 1.375
MEK ($\tau = t_f/2$)	1.012 / 2.445	1.283 / 3.458	0.393 / 1.028	0.220 / 0.504	0.445 / 1.125	0.179 / 0.415	0.380 / 0.718	0.512 / 1.102	0.901 / 1.935	0.592 / 1.414
MEK ($\tau = t_f$)	0.892 / 1.952	0.746 / 1.654	0.409 / 1.096	0.231 / 0.544	0.405 / 1.061	0.168 / 0.383	0.373 / 0.713	0.508 / 1.086	0.788 / 1.681	0.502 / 1.130
AML (K_0)	2.093 / 4.697	2.695 / 6.677	1.779 / 4.450	1.624 / 2.139	1.624 / 2.139	0.261 / 0.493	0.524 / 1.059	0.897 / 1.646	1.192 / 2.513	1.410 / 2.868
AML (*full*)	1.149 / 2.550	1.042 / 2.616	0.527 / 1.419	0.369 / 0.781	0.764 / 1.791	0.209 / 0.425	0.483 / 0.962	0.454 / 0.954	**0.414 / 0.974**	0.601 / 1.386
Ours	**0.537 / 1.137**	**0.391 / 0.824**	**0.259 / 0.613**	**0.155 / 0.316**	**0.336 / 0.807**	**0.155 / 0.325**	**0.323 / 0.656**	**0.357 / 0.770**	0.515 / 1.133	**0.336 / 0.731**

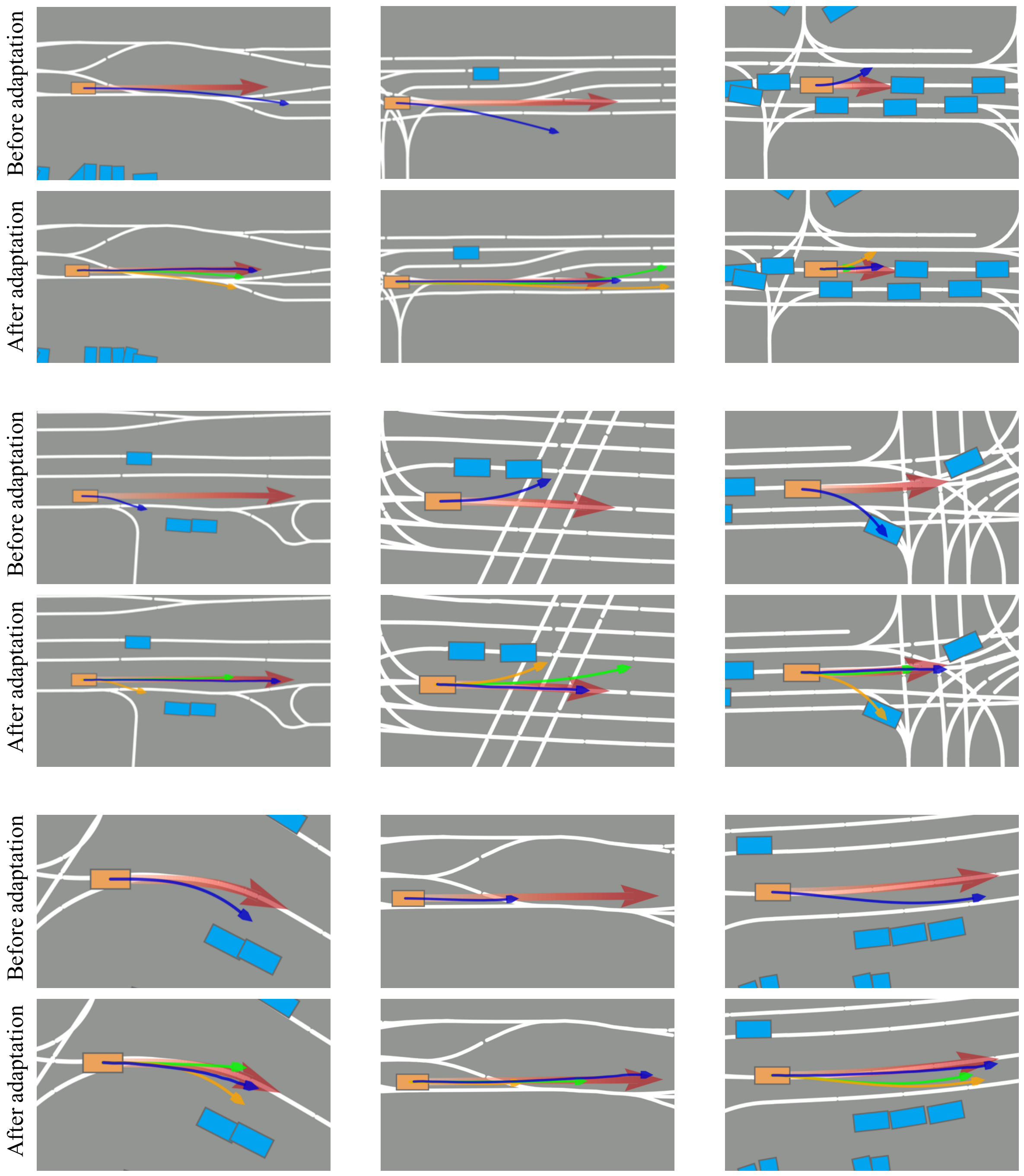

Figure 2. Additional visual comparison of adaptation results of our method against the baselines. For each image pair, the above shows prediction results before adaptation, and the below shows prediction results after adaptation via ours (blue arrow), TENT w/ sup (orange arrow), and MEK (green arrow). Red arrows denote GT future trajectories. Sky blue and orange boxes refer to surrounding actors and actors to be predicted. We depicted only one actor result and one mode among multi-modal predictions closest to the GT for visual simplicity. Please note that our method is multi-modal prediction for all actors method.

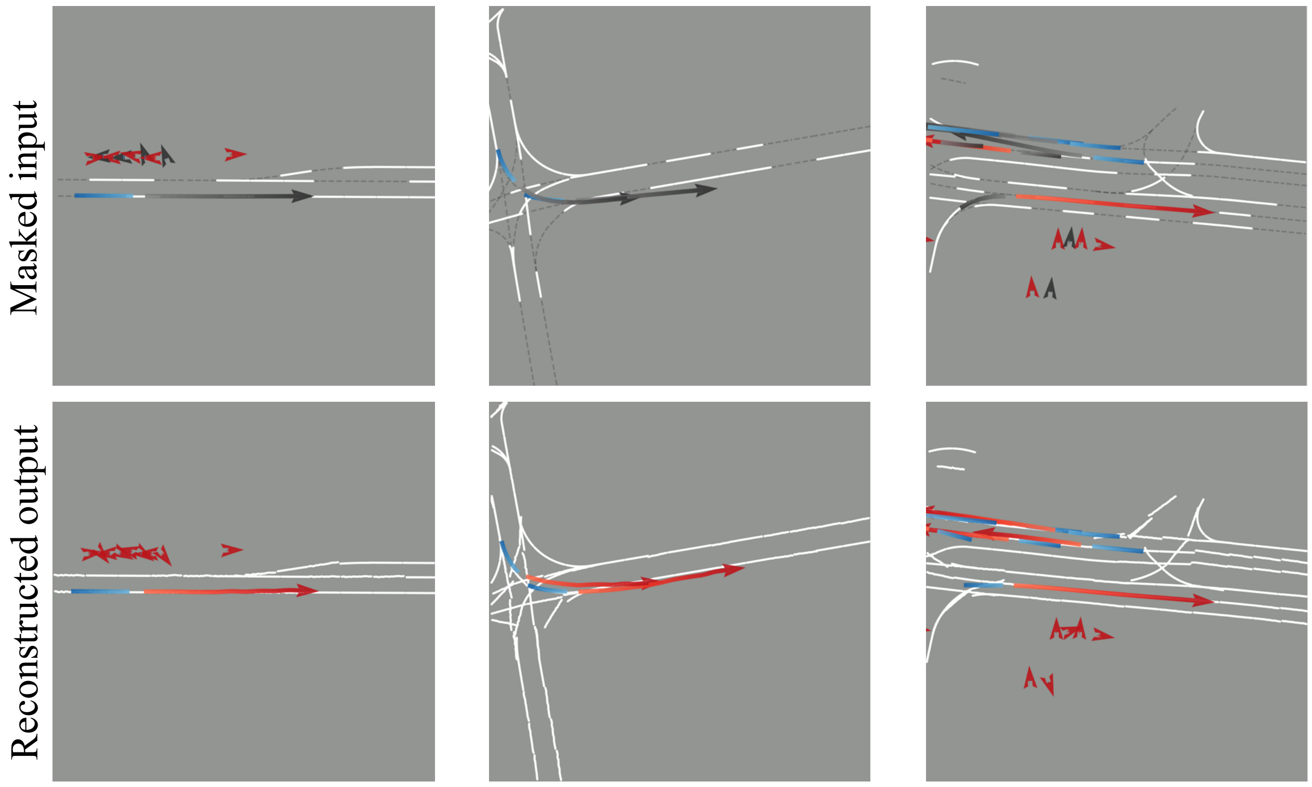

Figure 3. The first row indicates masked samples, and the row below shows the reconstructed outputs. The blue/red arrows indicate historical/future trajectories. The black arrows refer to the masked trajectories. The white lines are the lane centerlines, and the gray dashed lines are the masked lane centerlines.

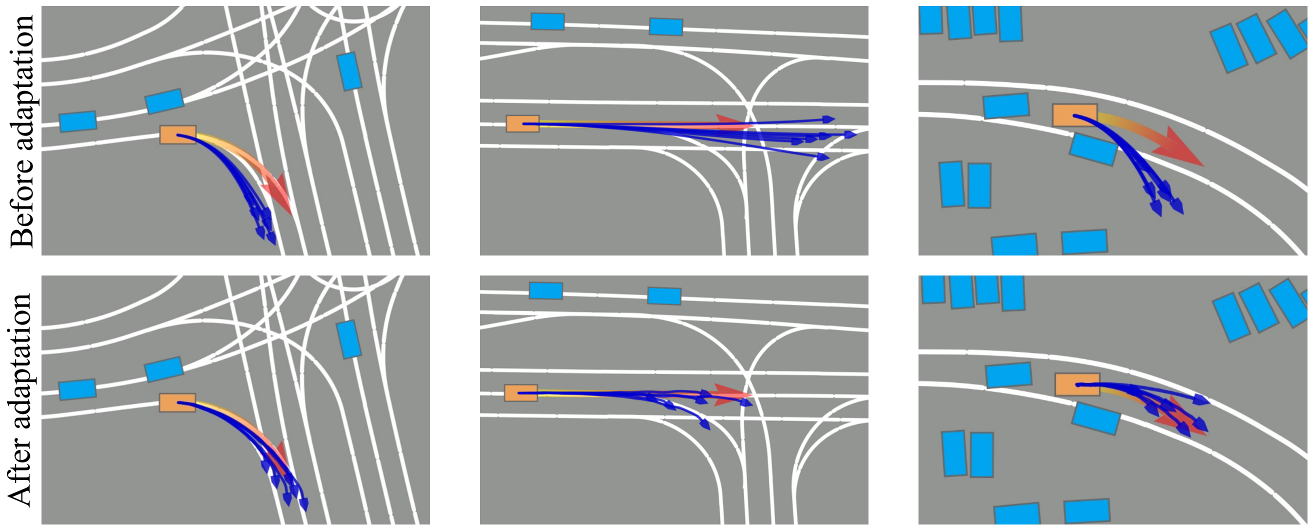

Figure 4. Multi modal prediction results (blue arrow) before and after adaptation via our method. Ours generates elaborate samples that consider interaction between lane or other actor due to representation learning, which cannot be learned from the GT (red arrow) using regression loss.