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

Daehee Park, Jaeseok Jeong, Sung-Hoon Yoon, Jaewoo Jeong, and Kuk-Jin Yoon
KAIST
{bag2824,jason.jeong,yoon307,jeong207,kjyoon}@kaist.ac.kr

Abstract

Trajectory prediction is a challenging problem that requires considering interactions among multiple actors and the surrounding environment. While data-driven approaches have been used to address this complex problem, they suffer from unreliable predictions under distribution shifts during test time. Accordingly, several online learning methods have been proposed using regression loss from the ground truth of observed data leveraging the auto-labeling nature of trajectory prediction task. We mainly tackle the following two issues. First, previous works underfit and overfit as they only optimize the last layer of motion decoder. To this end, we employ the masked autoencoder (MAE) for representation learning to encourage complex interaction modeling in shifted test distribution for updating deeper layers. Second, utilizing the sequential nature of driving data, we propose an actor-specific token memory that enables the test-time learning of actor-wise motion characteristics. Our proposed method has been validated across various challenging cross-dataset distribution shift scenarios including nuScenes, Lyft, Waymo, and Interaction. Our method surpasses the performance of existing state-of-the-art online learning methods in terms of both prediction accuracy and computational efficiency. The code is available at https://github.com/daeheepark/T4P.

1. Introduction

Trajectory prediction plays a significant role in autonomous systems, enhancing safety and navigation efficiency [28, 31]. Recently, data-driven methods have shown remarkable prediction capabilities [1, 16, 38, 52, 55, 57, 60, 81, 89, 90]; however, they are prone to distribution shift [9, 88]. Trajectory prediction models also produce unreliable output when faced shifts in the data distribution [25], posing significant risks in various real-world applications. This vulnerability stems from the ease with which the trajectory data distribution can be altered by numerous factors, such as scene changes and driving habits; *i.e.* road layout, interaction between agents, driver demographics [6, 51, 82].

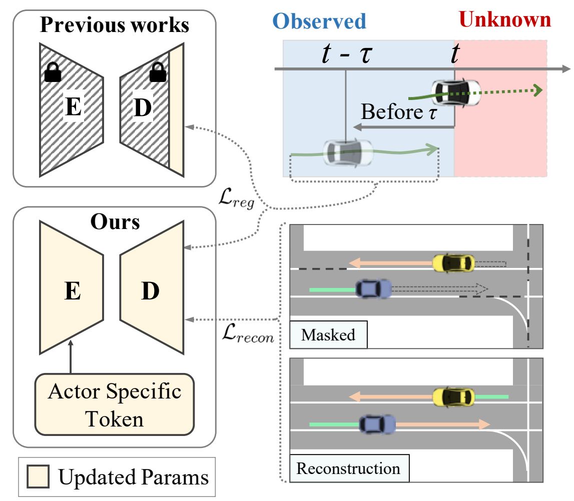

Figure 1. Previous methods optimize the last layer of the decoder using regression loss from delayed ground truth. Our method, on the other hand, learns representation via a masked autoencoder, which boosts prediction performance by optimizing deeper layers. In addition, the proposed actor-specific token enables the prediction model to learn actor-wise motion characteristics.

To address this challenge, recent methods have proposed domain adaptation and generalization strategies which aim to *anticipate* the distribution shifts and accordingly train the model [32, 76, 80]. However, due to the wide variety of factors influencing data distribution, the anticipated shifts may differ significantly from those encountered at test time. As a result, several online learning methods have been developed to dynamically adapt models during test time [36, 73]. Since trajectory prediction serves as an auto-labeling task where trajectory data is obtained from object tracking, the observed past and future trajectory provide both input and corresponding ground truth for supervision ($\mathcal{L}_{reg}$), as depicted at the top right of Fig. 1. Nevertheless, updating the entire model may ruin the representation learned from the source data, so only part of the network, such as the batch normalization layer, is updated [27, 40, 61, 83]. Particularly, because regression loss is calculated at delayed timestamps and only a few samples are available during test time, this approach risks deteriorating the model's learned representation [73]. Therefore, previous online prediction methods mainly update only the last layer of the decoder.

In this work, we propose a test-time training (TTT) for trajectory prediction with two key aspects. First, we build a masked autoencoder (MAE) framework to adapt deep features, incorporating good representation that captures complex interactions between agents and road structures. Due to the challenge of existing online learning methods in damaging learned representations when updating deeper layers, we employ a MAE to guide representation learning. Second, we introduce an actor-specific token memory that has significant advantages in real-world driving scenario where data arrives continuously and sequentially. As each actor instance has its own driving habits and the past motion pattern of specific actors can be accessed from the arrived observations, we design a token memory in transformer structure [68] and its training strategy to learn actor-wise motion characteristics. The proposed TTT framework is validated on challenging cross-dataset distribution shift cases between nuScenes, Lyft, Waymo, and INTERACTION, and shows state-of-the-art performance surpassing previous online learning methods. Furthermore, we show the practicality of our TTT framework by evaluating its computational efficiency. We summarize our contributions as below:

- We propose a test time training for trajectory prediction (T4P) by utilizing a masked autoencoder to learn deep feature representations that stably improve prediction performance across entire network layers.
- We introduce an actor-specific token memory used to learn the different actor characteristics and habits.
- Our method is validated across 4 different datasets as well as different temporal configurations. Ours achieves state-of-the-art performance both in accuracy and efficiency.

2. Related Works

2.1. Trajectory Prediction

Trajectory prediction garners attention with the emergence of methods that can enhance perception or planning [11, 12, 44]. Its goal is to predict future trajectories of traffic actors based on their historical trajectories and the context of their environment [3, 17, 19, 20, 39, 58, 63, 78, 91]. Historical trajectories, or tracklets of traffic actors, are sequentially acquired via vehicle detection and tracking systems. Some studies employed this temporal property to enhance prediction via memory replay [35, 42, 59]. In the early stages of trajectory prediction, only the historical trajectory of the actors of interest was considered. However, recent studies emphasize the significance of understanding interactions among agents [67, 85] and the rules governed by surrounding environments [72, 77] in improving prediction performance. This has led to the development of models that incorporate multi-head attention or graph-based methods to capture these interactions [24, 26, 42]. Additionally, MAE has been adopted for pretraining to better understand agent interactions [8, 13]. To further refine prediction capabilities, various generative models have been introduced, enabling the generation of future trajectories [15, 41, 71, 79].

2.2. Transfer Learning in Trajectory Prediction

With data-driven approaches offering superior performance in trajectory prediction, their effectiveness diminishes under distribution shifts [25, 56]. In response, several studies have adopted domain adaptation or generalization strategy [69, 74]. Some specifically aimed to reduce the domain gap within unique characteristics of trajectory prediction: differences in road structures [84], actor interaction [80], *etc*. However, these methods rely on anticipating how to cover domain shifts. Yet, given that trajectory data is subject to influence from numerous factors, predicting and accommodating for shifts may not always be sufficient. Consequently, recent developments have introduced adaptation to unseen test sets using online learning [33, 34, 43]. Among them, some methods [36, 73] showed remarkable prediction performance improvement under severe distribution shifts like cross-dataset cases by utilizing regression loss for online learning. These methods exploit the fact that the input and ground truth (GT) labels are provided at test time as tracking history. However, with the limitations of updating with only a few samples in a delayed time, adaptation becomes restricted to the last layer of the decoder.

2.3. Test Time Training

Test-time training (TTT) is a method that trains the network on test time data, unseen during training [4, 7, 14, 18, 22, 49, 66, 75]. Unlike domain generalization or adaptation, which are confined to the training phase, TTT extends model adaptation into the test phase by utilizing available test data [47]. TTT methods are categorized into regularization-based approaches for post-hoc regularization of out-of-distribution (OOD) samples [46, 87], and self-supervised approaches that employ pretext tasks on test data for optimal representation learning [7, 9, 45, 50, 54]. Specifically, *TTT* [65] introduced a Y-shaped network structure consisting of a feature encoder, a pretext branch, and a decoder branch. The decoder branch is fixed, while the encoder and pretext branch are optimized through self-supervision. Adhering to this model, *TTT-MAE* [23] integrated a MAE in the pretext task. Expanding on this method, we adopt *TTT-MAE* to the domain of trajectory prediction, leveraging its representation learning capabilities to enhance test-time training.

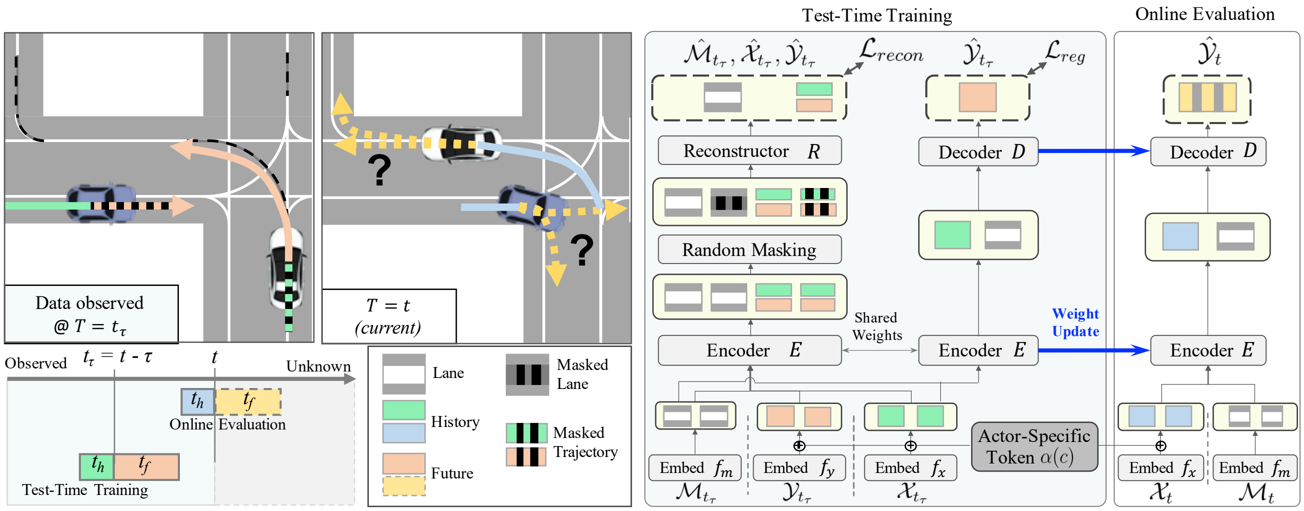

Figure 2. Overall method. During test-time training, the network trained on source dataset is optimized on target data under online setting. The model is optimized both from regression and reconstruction loss. Both losses utilize the data observed at the delayed time stamp (t_τ). Actor-specific token is used to learn instance-wise motion pattern during test-time training phase. During online evaluation phase, model and actor-specific token learned from test-time training phase are used.

3. Method

3.1. Problem definition

Trajectory prediction aims to learn the mapping function between the input, consisting of historical trajectory and map information $\mathbf{X} : \{\mathcal{X}_t, \mathcal{M}_t\}$, and the output, consisting of K possible candidates for future trajectory of N traffic actors, $\mathbf{Y} : \{\mathcal{Y}_t^{0:K-1}\}$, at current time t. We predict C different actor classes including vehicle, cyclist, *etc*. Historical and future trajectories are represented as $\mathcal{X}_t = \mathbf{x}_{t-t_h:t}^{0:N-1}$ and $\mathcal{Y}_t = \mathbf{x}_{t:t+t_f}^{0:N-1}$ where t_h and t_f represent sequence length of input and output trajectory. Here, $\mathbf{x}_t^n$ represents the spatial location of actor n at time t. For map information $\mathcal{M}_t$, we use L segmented lane centerlines around ego-actors which is widely-used in trajectory prediction methods.

We deal with the case when the target data distribution during test time $\{\mathbf{X}, \mathbf{Y}\}^T$ is different from the source data distribution seen during the training phase $\{\mathbf{X}, \mathbf{Y}\}^S$. We formulate the problem as a online adaptation scenario in which one data sample is given per each time interval as time passes. The test data is consists of multiple distinct *scenes*. Each scene includes *temporally ordered* data samples which are captured through real-world driving. Following standard TTT methods, there is no access to the source data at test time. However, thanks to the auto-labeling nature of trajectory prediction, there is access to delayed GT future trajectory ($\mathbf{x}_{t_\tau:t_\tau+t_f}$) from a previous time window at time $t_\tau(= t - \tau)$ as depicted in left lower corner of Fig. 2.

3.2. Overall method

Our method, Test-Time Training of Trajectory Prediction (T4P), enhances the online learning method using supervision from a delayed GT future trajectory with representation learning from MAE and actor-specific token memory. Following standard *TTT* [65], the overall framework consists of three phases: *offline training*, *test-time training*, and *online evaluation*. Offline training occurs before test time, and test-time training and online evaluation are executed repeatedly and sequentially during test time. We adopt the ForecastMAE [13] backbone consisting of embedding layers f, a shared encoder E, a reconstruction head R and a motion decoder head D, as depicted in the middle of Fig. 2. The detailed methods during each phase are described below:

3.3. Offline training

During offline training, the model is trained on source data using both reconstruction loss and regression loss.

$$\min_{\theta \in \{f,E,R,D\}} \mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{M} \in \{\mathbf{X},\mathbf{Y}\}^S} \left[\mathcal{L}_{recon} + \mathcal{L}_{reg}\right] \tag{1}$$

In this subsection, subscript t is omitted for simplicity. First, all input elements ($\mathcal{X}, \mathcal{Y}, \mathcal{M}$) are embedded with their respective embedding layer (f_x, f_y, f_m). Additionally, we define actor class token $\bar{\alpha} \in \mathbb{R}^{C \times D}$ that learns different motion patterns of each actor class. The actor class token is implemented as a learnable embedding of a transformer structure. Corresponding actor class token $\alpha(c)$ is added to trajectory embedding according to the class of each actor.

$$h_x, h_y, h_m = f_x(\mathcal{X}) + \alpha(c), f_y(\mathcal{Y}) + \alpha(c), f_m(\mathcal{M}) \tag{2}$$

For reconstruction, the history/future trajectory and lane embeddings are fed to the encoder to obtain the encodings F_x, F_y, F_m. Then, segments of the encodings are randomly masked and replaced with masking tokens, M_x, M_y, M_m, and the other encodings remain unmasked (F'_x, F'_y, F'_m). Here, we use random masking for the lane centerline and complementary masking strategy for history/future trajectory following previous works [13]. The masking tokens and the unmasked encodings are fed to the reconstructor to reconstruct the masked elements.

$$
\begin{aligned}
&F_x, F_y, F_m = E(h_x, h_y, h_m) \\
&\hat{\mathcal{X}}, \hat{\mathcal{Y}}, \hat{\mathcal{M}} = R(M_x, M_y, M_m, F'_x, F'_y, F'_m)
\end{aligned} \tag{3}
$$

The encoder and reconstructor both consist of multi-head attention to utilize interaction between history, future and lanes, thus, the reconstruction guides to the model have the capability of interaction reasoning. Finally, a reconstruction loss $\mathcal{L}_{recon}$ is computed as the MSE loss between the ground truth and the reconstructed outputs.

$$
\mathcal{L}_{recon} = \frac{1}{N}\sum_n(\mathcal{X}-\hat{\mathcal{X}})^2 + \frac{1}{N}\sum_n(\mathcal{Y}-\hat{\mathcal{Y}})^2 + \frac{1}{L}\sum_l(\mathcal{M}-\hat{\mathcal{M}})^2 \tag{4}
$$

For the decoder head, historical trajectory and lanes embeddings are again fed to the same encoder and the output encodings are passed to the motion decoder, composed of MLP layers. The decoder outputs K candidates for trajectory prediction, and the regression loss $\mathcal{L}_{reg}$ is computed with the widely-used Winner-takes-all (WTA) loss [29, 48].

$$
\begin{aligned}
&\hat{\mathcal{Y}}^{0:K-1} = D(E(h_h, h_l)) \\
&\mathcal{L}_{reg} = \frac{1}{N}\sum_n \underset{k\in K}{\text{argmin}}(\mathcal{Y}^n - \hat{\mathcal{Y}}^{n,k})^2
\end{aligned} \tag{5}
$$

3.4. Test-time training

During test-time, a data sample consisting of trajectories and maps arrives sequentially. Therefore, even though we cannot access the GT future trajectory of current time ($\mathcal{Y}_t$), we can access both the inputs and GT ($\mathcal{X}_{t_\tau}, \mathcal{Y}_{t_\tau}, \mathcal{M}_{t_\tau}$) at a previous time t_τ. With this data, the model is optimized with the same objective as Eq. 1 with target data distribution instead of source data distribution.

$$
\min_{\theta\in\{f,E,R,D\}} \mathbb{E}_{\mathcal{X}_{t_\tau}, \mathcal{Y}_{t_\tau}, \mathcal{M}_{t_\tau}\in\{\mathbf{X},\mathbf{Y}\}^T}\left[\mathcal{L}_{recon} + \mathcal{L}_{reg}\right] \tag{6}
$$

Unlike existing online learning methods that only utilize regression loss, we incorporate an additional reconstruction loss. This enables the model to learn a good representation that considers the complex actor-actor and actor-lane interaction even in the unseen target data distribution. An advantage of representation learning is that the performance stably improves even when the deeper layers are optimized.

3.4.1 Actor-specific token memory

Unlike during the offline training phase, when the data order is shuffled, data at test-time comes in sequentially. Therefore, at test time, it is possible to keep track of movement patterns of a specific actor instance. Using this, we propose an actor-specific token memory.

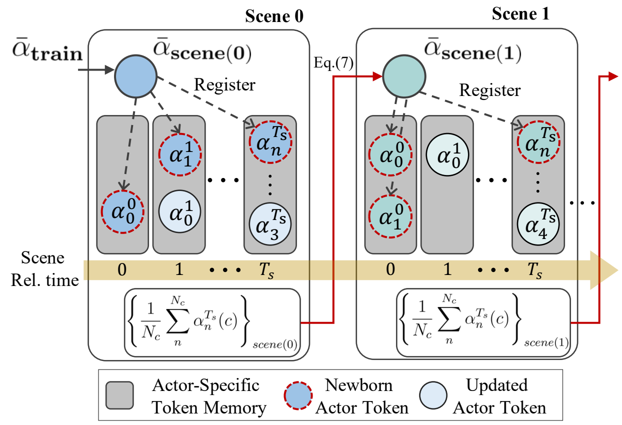

Figure 3. Actor-specific token memory is colored in gray. It evolves as time passes within a scene. For newborn actors, the corresponding class token is registered. Until the actor disappears, the token is updated through test-time training. At the end of the scene, all tokens are averaged by each class and passed to the next scene as denoted in red arrow and Eq. 7.

The overall scheme is described in Fig. 3. During offline training, actor class tokens $\bar{\alpha}_{train} \in \mathbb{R}^{C\times D}$ are trained to reflect the average motion pattern of each of the C classes. At the beginning of the test-time, scene 0, class tokens are initialized from that of training phase ($\bar{\alpha}_{scene(0)} \leftarrow \bar{\alpha}_{train}$). When a new n^{th}-actor appears at time t, the class token $\alpha^t_n(c)$ is cloned from $\bar{\alpha}_{scene(0)}$ by selecting corresponding class. The newborn tokens are then registered to the actor-specific token memory. The token memory is structured as a dictionary where actor instance ID/corresponding tokens are key/values. At each iteration, the actor-specific tokens are used for both test-time training and online evaluation. As time progresses, the actor-specific tokens evolve and are updated through the reconstruction and regression losses until the actor disappears in the scene. By giving each actor its own specific token that distinguishes it from the sharing of other parts of models with other actors, actor-specific motion patterns can be learned.

When the scene changes, scene 1, the actors observed during scene 0 are not to be observed anymore, so we need a new averaged actor class token $\bar{\alpha}_{scene(1)}$. For that, we average all the tokens in the memory at the final time step T_S of scene 0 after gathering by classes as Eq. 7. Here, N_c refers to the number of actors of class c. It is because, with a sufficient number of actors class tokens per each class in memory, their average motion can be a representative motion pattern of actor classes. This is more useful than $\bar{\alpha}_{train}$ because newly averaged tokens are trained on the target dataset while $\bar{\alpha}_{train}$ contains motion pattern trained on the source dataset. The averaged class tokens are then passed to the next scene and used to initialize tokens for the

Table 1. Adaptation results in various distribution shifts. The model is trained on source dataset, and test-time trained and evaluated on target dataset (*Source* $\rightarrow$ *Target*). All metrics are better in lower value. The best and second-best results are marked in **bold** and underline.

mADE_6 / mFDE_6	Short-term exp (1/3/0.1)				Long-term exp (2/6/0.5)			
	INTER $\rightarrow$ nuS	INTER $\rightarrow$ Lyft	nuS $\rightarrow$ Way	Mean	nuS $\rightarrow$ Lyft	Way $\rightarrow$ Lyft	Way $\rightarrow$ nuS	Mean
Source Only	1.047 / 2.247	1.391 / 2.945	0.431 / 1.031	0.956 / 2.074	1.122 / 2.577	0.621 / 1.347	1.153 / 2.220	0.965 / 2.048
Joint Training	1.116 / 2.445	1.553 / 3.458	0.472 / 1.125	1.047 / 2.343	1.108 / 2.597	0.638 / 1.404	1.091 / 2.031	0.946 / 2.011
DUA	1.118 / 2.455	1.516 / 3.352	0.516 / 1.294	1.050 / 2.367	1.365 / 3.257	0.790 / 1.868	1.270 / 2.634	1.142 / 2.586
TENT (w/ sup)	1.102 / 2.423	1.519 / 3.405	0.448 / 1.071	1.023 / 2.300	1.068 / 2.514	0.628 / 1.381	1.077 / 2.012	0.924 / 1.969
MEK ($\tau = t_f/2$)	1.012 / 2.445	1.283 / 3.458	0.445 / 1.125	0.913 / 2.343	1.079 / 2.597	0.629 / 1.404	1.079 / 2.031	0.929 / 2.011
MEK ($\tau = t_f$)	0.892 / 1.952	0.746 / 1.654	0.405 / 1.061	0.691 / 1.556	1.006 / 2.369	0.615 / 1.351	1.117 / 2.140	0.913 / 1.953
AML (K_0)	2.093 / 4.697	2.695 / 6.677	1.624 / 2.139	2.137 / 4.504	1.787 / 3.067	1.322 / 2.571	1.618 / 2.999	1.576 / 2.879
AML (*full*)	1.149 / 2.550	1.042 / 2.616	0.764 / 1.791	0.985 / 2.319	1.462 / 2.573	0.977 / 2.184	1.495 / 2.978	1.311 / 2.578
Ours (T4P)	**0.537 / 1.137**	**0.391 / 0.824**	**0.336 / 0.807**	**0.421 / 0.923**	**0.776 / 1.820**	**0.549 / 1.171**	**0.996 / 1.784**	**0.774 / 1.592**

newborn actors. Please note that while scene 0 is initialized by actor class tokens from the training phase, subsequent scenes obtain as in Eq. 7. More details of memory evolving strategies can be found on the supplementary material.

$$\bar{\alpha}_{scene(i+1)} \leftarrow \left\{ \frac{1}{N_c} \sum_{n}^{N_c} \alpha_n^T(c) \right\}_{scene(i)} \quad (7)$$

3.5. Online evaluation

Using the updated model weight and actor-specific token memory during test-time training, online evaluation is executed. With the input data $(\mathcal{X}_t, \mathcal{M}_t)$ at current time t, the learned encoder and decoder predict multi-modal trajectory $(\mathcal{Y}_t)$ for all actors in the sample.

4. Experiment

4.1. Datasets

We conducted experiments on well-known datasets, nuScenes [5], Lyft [30], WOMD [21], and INTERACTION [86], to evaluate T4P on various data distribution shifts. These datasets are parsed into the same format using *trajdata* [37]. Additionally, to verify in various prediction configurations, experiments were conducted with the two most widely used configurations of long-term and short-term prediction. Long-term prediction requires predicting 6 seconds into the future given 2 seconds of the past with a time interval of 0.5s, making the input/output sequence lengths to be 5 and 12, respectively. Short-term prediction requires predicting 3 seconds into the future given 0.9 seconds of the past with a time interval of 0.1s, making the input/output sequence lengths to be 10 and 30, respectively.

4.2. Implementation details

For actor classes, we use the 5 classes: *unknown*, *vehicle*, *pedestrian*, *bicycle* and *motocycle*. Our method predicts K=6 future candidates for all actors in the sample. We use τ as t_f to enable the past GT future to contain full prediction horizon. We train and evaluate our model with a single NVIDIA A6000. Learning rates of model weight and actor-specific parameters are set as 0.01 and 0.5, respectively, and weight decay is set to 0.001 for all. The gradient is clipped by 15. For metrics, widely used mADE_6 and mFDE_6 are used. Detailed metric definition, model architecture, and training details are included in the supplementary material.

4.3. Baselines

We compare our *T4P* with several baselines, including unsupervised/supervised test-time-training methods and online learning trajectory prediction methods. All baseline methods are implemented using the same backbone.

Source only refers to the backbone model trained on the source dataset only using regression loss.

Joint training is similar to source only but trained with regression and reconstruction loss jointly.

DUA [53] is an unsupervised post-hoc regularization method only updates batch normalization statistics in a momentum-updating manner without back-propagation.

TENT with supervision is a variant of the original TENT [70] in which regression loss is used to optimize the batch normalization layers instead of entropy minimization loss, as entropy minimization is not applicable.

MEK [73] is an online learning trajectory prediction method utilizing the Modified Extended Kalman filter. It uses only regression loss to optimize the last layer of the decoder. As the prediction horizon is different in our experiment from the original paper, we use both $\frac{1}{2}t_f$ and t_f.

AML [36] is an Adaptive Meta-learning method. Unlike the other methods that use the same backbone, AML replaces the last decoder layer with a Bayesian linear regression layer for adaptive training. The modified version of backbone without adaptive training is denoted as K_0, while the full version with adaptive training is denoted as *full*.

5. Results

5.1. Quantitative results

The results of comparing our method with the baselines in various distribution shift scenarios are presented in Tab. 1. We reported three distribution shift scenarios per each time configuration on the table, and other results are included in

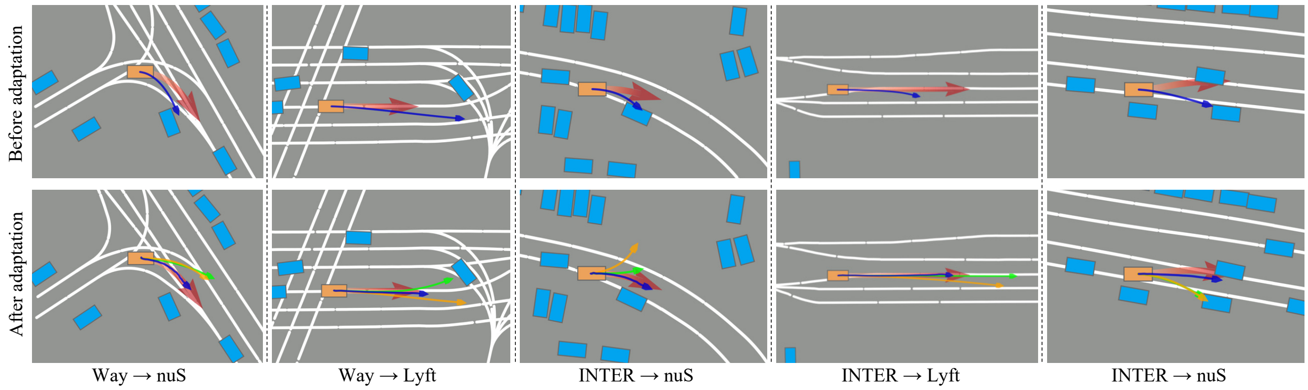

Figure 4. The first row shows prediction before adaptation, and the second row indicates adaptation results by three methods: ours (blue), TENT w/ sup (orange) and MEK (green). Sky blue and orange boxes refer to surrounding actors and actors to be predicted. We depicted only one actor result and one 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.

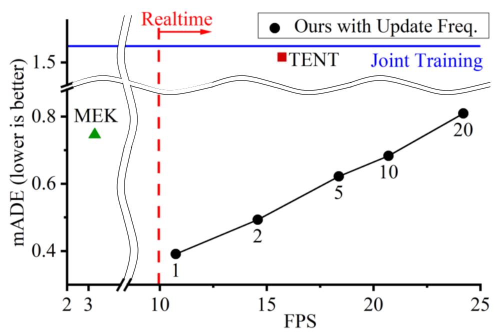

Figure 5. Prediction accuracy and execution time on INTER → nuS (1/3/0.1) experiment. Adjusting update frequency can balance between accuracy and efficiency. Our method significantly outperforms the baseline methods in both accuracy and efficiency.

the supplementary material. Notably, our approach consistently surpassed baseline performance across all scenarios.

DUA consistently exhibits compromised performance across nearly all cases, due to the distinctive characteristics of trajectory data. In contrast to image data, where data is treated as a singular sample, trajectory data involves multiple agents, each exhibiting distinct motion patterns, within a single data. Consequently, holistically updating batch statistics proves to be counterproductive. Similar challenges are encountered by TENT w/ sup. While regression loss prevents performance decline, updating only the batch norm layer has little to no effect on the prediction performance.

MEK exhibited the most competitive performance among the baselines. While performance improved significantly in short-term settings, it brings limited improvement in long-term cases. As the Kalman filter updates based on the number of prediction steps, short-term configurations with 12 update steps show a better improvement than long-term configurations with only 5 update steps.

Although AML led to a considerable improvement in the *full* version, the predictive performance itself was substantially degraded due to the significant performance drop in the modified backbone (K_0). The limitation of the backbone is due to the Bayesian regression layer being based on probability sampling which is known to be worse than non-probability sampling method of ours [2]. In contrast to all the baseline methods, our method demonstrated state-of-the-art performance in all scenarios featuring various distribution shifts, whether short-term or long-term, showcasing the generalizability of our approach.

5.1.1 Efficiency

As efficiency is a crucial factor in TTT, we evaluate the frame per second (FPS) along with accuracy (mADE$_6$), shown in Fig. 5. We set the performance of the joint training method w/o adaptation as the benchmark and present MEK and TENT, which demonstrates competitive performance among the baselines. Our approach allows for the adjustment of the update frequency, with a frequency of 1 indicates updating at every opportunity, and 2 means updating every other opportunity. While frequent updates improve prediction performance, they also increase execution time; adjusting the update frequency allows for a balance between efficiency and accuracy. As shown in Fig. 5, our method outperforms in both accuracy and efficiency. Given that the time interval is 0.1 seconds, real-time execution requires a processing speed of at least 10 FPS. Even at the maximum update frequency of 1, our method maintains real-time capability with a superior accuracy of 0.39. When increasing the update frequency to 20, the error increases to 0.81, close to MEK's 0.75. However, the FPS reaches 24.2, demonstrating overwhelmingly faster operation compared to MEK's speed of 3.3 FPS.

5.2. Qualitative results

Comparison to the baselines: We compare our results with TENT and MEK in Fig. 4. While all methods, includ-

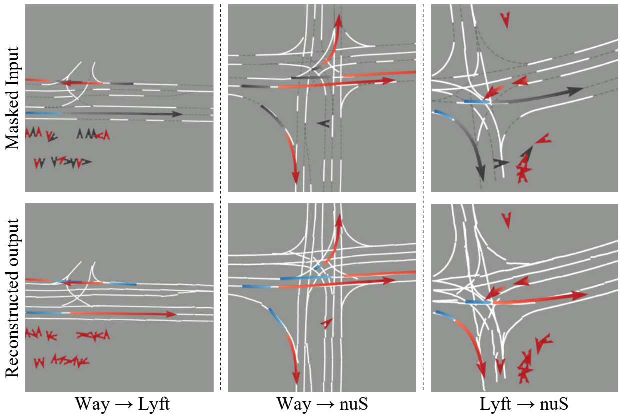

Figure 6. The first row indicates masked samples, and the row below shows the reconstructed outputs. The blue/red arrows indicate historical/future trajectories. The black arrows refer to the masked trajectories. The white lines are the lane centerlines, and the gray dashed lines are the masked lane centerlines.

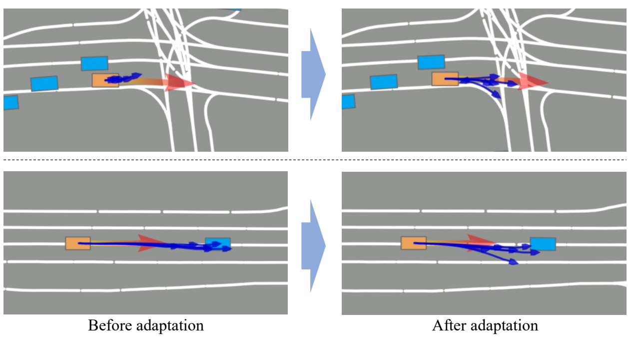

Figure 7. Multi modal prediction results (blue arrows) before and after adaptation via our method. Ours generates elaborate samples that consider interaction between lane (above) or actors (below) due to representation learning, which cannot be learned from the GT (red arrow) via regression loss.

ing ours, perform multi-agent, multi-modal predictions, we only illustrate one actor and the closest mode to the GT for visual simplicity. The first row of the figure represents predictions before adaptation, and below are the results after adaptation using three different methods. MEK and TENT exhibit instances of underfitting or excessive overfitting upon adaptation, whereas our method consistently demonstrates stable and accurate predictions.

Reconstruction results: Reconstruction examples are depicted in Fig. 6. For all agents and lanes within a data sample, random masking is applied, as shown in the first row. During test-time training, learning for reconstruction is conducted, resulting in successful reconstruction for data with different distributions, as seen in the second row.

Multi-modal prediction results: As multi-modality is a crucial issue [10, 62, 64], we show that ours can handle multi-modal prediction results in Fig. 7. The adapted predictions showcase diverse yet plausible scenarios, either considering the lane structure (above) or surrounding agents (below). These elaborated samples, although not present in the observed ground truth (GT) future, are learned through the representation learning from reconstruction loss. In addition, it shows that actor-specific tokens do not induce mode collapse to only one motion.

Table 2. Effect of type of losses to be optimized. Optimizing all losses shows optimal test-time adaptation performance.

Exp.	Loss type			mADE_6 /mFDE_6
	Actor recon	Lane recon	Reg	
INTER → Lyft (1/3/0.1)				1.553 / 3.458
	✓			1.054 / 2.007
	✓	✓		0.842 / 1.512
			✓	0.674 / 1.430
	✓	✓	✓	**0.391 / 0.824**
nuS → Lyft (2/6/0.5)				1.108 / 2.597
	✓			0.987 / 2.304
	✓	✓		0.973 / 2.280
			✓	0.942 / 2.262
	✓	✓	✓	**0.776 / 1.820**

Table 3. Ablation on the depth of optimizing layer according to the loss types. The right side of the table indicates the optimization of deeper layers. Using only $\mathcal{L}_{reg}$ deteriorates performance when optimizing all layers while ours stably improves as deeper.

Loss	Optimizing layers		
	D	D+E	D+E+$f_{h,f,l}$
$\mathcal{L}_{reg}$	0.864 / 2.072	**0.840 / 2.093**	0.942 / 2.262
$\mathcal{L}_{reg}+\mathcal{L}_{recon}$	0.859 / 2.060	0.813 / 1.923	**0.776 / 1.820**

6. Ablation

6.1. Reconstruction objective

Table. 2 shows ablation studies on optimizing different loss types. Both reconstruction and regression losses individually boost prediction performance, with their joint optimization yielding even greater improvements. Table. 3 compares the effects of using only regression loss versus both losses on prediction performance across different layer depths. Updating just the decoder (D) shows similar results in both scenarios, but extending updates to the encoder (E) significantly enhances performance when using both losses. Furthermore, extending updates to the embedding layers ($f_{h,f,l}$) deteriorates performance when only regression loss is optimized. This highlights the importance of incorporating representation learning through the MAE, as relying solely on regression loss can lead to suboptimal adaptation and damage to learned representations.

We also conduct ablation studies on the masking ratio for both actors and lane centerlines in Tab. 4. The result is visualized via graphs in Fig. 8 according to lane masking ratio and actor masking ratio, respectively. Around 0.3 of lane and 0.4 of actor masking ratio, tendencies of mADE_6 follow

Table 4. mADE$_6$ according to actor and lane masking ratio.

INTER → Lyft (1/3/0.1)		Lane Masking Ratio 0.1	0.3	0.5	0.7	0.9
Actor Masking Ratio	0.1	0.418	0.407	0.464	0.481	0.446
	0.3	0.423	0.443	0.443	0.445	0.390
	0.5	0.417	0.455	0.515	0.391	0.435
	0.7	0.567	0.435	0.448	0.453	0.491
	0.9	0.447	0.421	0.494	0.417	0.398

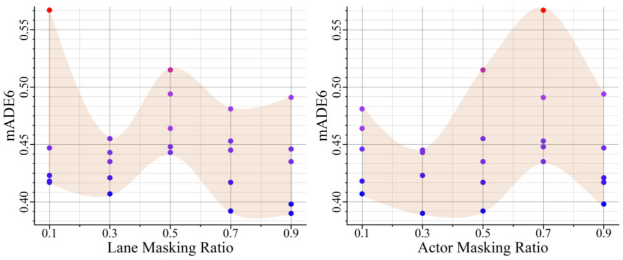

Figure 8. Tendency of mADE$_6$ according to actor and lane masking ratio respectively. (INTER → Lyft (1/3/0.1))

a U-shape. In case of too small masking ratio, the reconstruction does not learn sufficient representation from the loss, while large masking ratio interrupt interaction learning due to absence of sufficient information. However, in both lane and actor masking, mADE$_6$ gets improved when the masking ratio increases above 0.8. In that case, the reconstruction network is induced to learn scene-specific information. In addition, unlike regression loss which deteriorates performance, reconstruction loss does not harm performance because it induces learning the semantic relationship than direct regression supervision.

6.2. Actor-specific token

Table. 5 presents results for the baseline without adaptation, our method without actor-specific tokens, and our full method. The second column reveals that even without actor-specific tokens, the prediction performance is 0.581 and 0.931, surpassing MEK's 0.746 and 1.006. However, incorporating actor-specific tokens for instance-aware adaptation yields notable improvements of 32.7% and 16.7% for short-term and long-term experiments, respectively. The difference in performance between short-term and long-term is influenced by the scene length in the dataset. The average scene length for short-term data with a 0.1 time interval is 200.04, significantly longer than the average of 32.67 for long-term data with a 0.5 time interval. Intuitively, as scene length increases, the time spent observing previously adapted actors also increases, enhancing the effectiveness of actor-specific tokens. To verify this, Fig. 9 adjusts scene length arbitrarily by skipping to the next scene in data loading once a specific scene length is exceeded. The results confirm that as scene length decreases by skipping scenes earlier, the effectiveness of actor-specific tokens diminishes in both short-term and long-term scenarios.

Table 5. Effect of actor-specific token in mADE$_6$/mFDE$_6$. The proposed method enhances adaptation performance by learning actor-wise motion characteristics.

Exp.	Baseline	Ours w/o Actor-specific	Ours (Full)
INTER → Lyft (1/3/0.1)	1.553 / 3.458	0.581 / 1.151	**0.391 / 0.824**
nuS → Lyft (2/6/0.5)	1.108 / 2.597	0.932 / 2.220	**0.776 / 1.820**

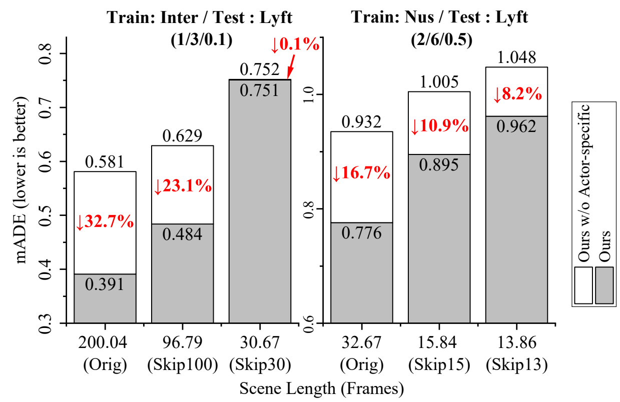

Figure 9. Effect of scenario length to the effectiveness of actor-specific token. As the scenario length shortens with manual skipping, its effectiveness diminishes because the duration available for the actor-specific token to adapt is reduced. In real-world applications, driving scenarios are continuous, resulting in maximal efficacy of the proposed method.

7. Conclusion

We propose a test-time training method for trajectory prediction by incorporating the MAE and actor-specific token memory. The introduced MAE objective addresses a limitation of conventional online learning, preventing the loss of representations learned from source data. Consequently, our approach enables learning deeper layers, leading to improved representations and enhanced predictions even for out-of-distribution samples. The integration of actor-specific tokens during test-time allows for instance-wise learning of motion patterns, resulting in substantial performance improvements. This approach, particularly effective in continuous real-world autonomous driving scenarios without scene breaks, demonstrates significant efficacy and holds promise for practical applications.

Acknowledgements This research was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF2022R1A2B5B03002636) and the Challengeable Future Defense Technology Research and Development Program through the Agency For Defense Development (ADD) funded by the Defense Acquisition Program Administration (DAPA) in 2024 (No.912768601).

References

[1] Görkay Aydemir, Adil Kaan Akan, and Fatma Güney. Adapt: Efficient multi-agent trajectory prediction with adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8295–8305, 2023. 1

[2] Inhwan Bae, Jin-Hwi Park, and Hae-Gon Jeon. Non-probability sampling network for stochastic human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6477–6487, 2022. 6

[3] Inhwan Bae, Jean Oh, and Hae-Gon Jeon. Eigentrajectory: Low-rank descriptors for multi-modal trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10017–10029, 2023. 2

[4] Collin Burns and Jacob Steinhardt. Limitations of post-hoc feature alignment for robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2525–2533, 2021. 2

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 5

[6] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8483–8492, 2019. 1

[7] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022. 2

[8] Hao Chen, Jiaze Wang, Kun Shao, Furui Liu, Jianye Hao, Chenyong Guan, Guangyong Chen, and Pheng-Ann Heng. Traj-mae: Masked autoencoders for trajectory prediction. *arXiv preprint arXiv:2303.06697*, 2023. 2

[9] Liang Chen, Yong Zhang, Yibing Song, Ying Shan, and Lingqiao Liu. Improved test-time adaptation for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24172–24182, 2023. 1, 2

[10] Weihuang Chen, Zhigang Yang, Lingyang Xue, Jinghai Duan, Hongbin Sun, and Nanning Zheng. Multimodal pedestrian trajectory prediction using probabilistic proposal network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022. 7

[11] Xuesong Chen, Shaoshuai Shi, Chao Zhang, Benjin Zhu, Qiang Wang, Ka Chun Cheung, Simon See, and Hongsheng Li. Trajectoryformer: 3d object tracking transformer with predictive trajectory hypotheses. *arXiv preprint arXiv:2306.05888*, 2023. 2

[12] Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Scept: Scene-consistent, policy-based trajectory predictions for planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17103–17112, 2022. 2

[13] 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. 2, 3, 4

[14] Junhyeong Cho, Gilhyun Nam, Sungyeon Kim, Hunmin Yang, and Suha Kwak. Promptstyler: Prompt-driven style generation for source-free domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15702–15712, 2023. 2

[15] Dooseop Choi and KyoungWook Min. Hierarchical latent structure for multi-modal vehicle trajectory forecasting. In *European Conference on Computer Vision*, pages 129–145. Springer, 2022. 2

[16] Sehwan Choi, Jungho Kim, Junyong Yun, and Jun Won Choi. R-pred: Two-stage motion prediction via tube-query attention-based trajectory refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8525–8535, 2023. 1

[17] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 2

[18] Ning Ding, Yixing Xu, Yehui Tang, Chao Xu, Yunhe Wang, and Dacheng Tao. Source-free domain adaptation via distribution estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7212–7222, 2022. 2

[19] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020. 2

[20] Yonghao Dong, Le Wang, Sanping Zhou, and Gang Hua. Sparse instance conditioned multimodal trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9763–9772, 2023. 2

[21] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021. 5

[22] Francois Fleuret et al. Uncertainty reduction for model adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9613–9623, 2021. 2

[23] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022. 2

[24] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. THOMAS: Trajectory heatmap output with learned multi-agent sampling. In *In-*

ternational Conference on Learning Representations, 2022. 2

[25] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Uncertainty estimation for cross-dataset performance in trajectory prediction. *arXiv preprint arXiv:2205.07310*, 2022. 1, 2

[26] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations*, 2022. 2

[27] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. NOTE: Robust continual test-time adaptation against temporal correlation. In *Advances in Neural Information Processing Systems*, 2022. 2

[28] Junru Gu, Chenxu Hu, Tianyuan Zhang, Xuanyao Chen, Yilun Wang, Yue Wang, and Hang Zhao. Vip3d: End-to-end visual trajectory prediction via 3d agent queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5496–5506, 2023. 1

[29] 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. 4

[30] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *Conference on Robot Learning*, pages 409–418. PMLR, 2021. 5

[31] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023. 1

[32] Pingxuan Huang, Zhenhua Cui, Jing Li, Shenghua Gao, bo Hu, and Yanyan Fang. Cross-domain trajectory prediction with ctp-net, 2022. 1

[33] Manh Huynh and Gita Alaghband. Aol: Adaptive online learning for human trajectory prediction in dynamic video scenes. *arXiv preprint arXiv:2002.06666*, 2020. 2

[34] Manh Huynh and Gita Alaghband. Online adaptive temporal memory with certainty estimation for human trajectory prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 940–949, 2023. 2

[35] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[36] Boris Ivanovic, James Harrison, and Marco Pavone. Expanding the deployment envelope of behavior prediction via adaptive meta-learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7786–7793. IEEE, 2023. 1, 2, 5

[37] 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. 5

[38] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023. 1

[39] Ruochen Jiao, Xiangguo Liu, Takami Sato, Qi Alfred Chen, and Qi Zhu. Semi-supervised semantics-guided adversarial training for robust trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8207–8217, 2023. 2

[40] Junho Kim, Inwoo Hwang, and Young Min Kim. Ev-tta: Test-time adaptation for event-based object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17745–17754, 2022. 2

[41] Mihee Lee, Samuel S Sohn, Seonghyeon Moon, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. Muse-vae: multi-scale vae for environment-aware long term trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2221–2230, 2022. 2

[42] Lihuan Li, Maurice Pagnucco, and Yang Song. Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2231–2241, 2022. 2

[43] Maosen Li, Siheng Chen, Yanning Shen, Genjia Liu, Ivor W Tsang, and Ya Zhang. Online multi-agent forecasting with interpretable collaborative graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[44] Yingwei Li, Charles R Qi, Yin Zhou, Chenxi Liu, and Dragomir Anguelov. Modar: Using motion forecasting for 3d object detection in point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9329–9339, 2023. 2

[45] Yushu Li, Xun Xu, Yongyi Su, and Kui Jia. On the robustness of open-world test-time training: Self-training with dynamic prototype expansion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11836–11846, 2023. 2

[46] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pages 6028–6039. PMLR, 2020. 2

[47] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023. 2

[48] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 541–556. Springer, 2020. 4

[49] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. Ttn: A domain-shift aware batch normalization in test-

time adaptation. In *The Eleventh International Conference on Learning Representations*, 2022. 2

[50] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34: 21808–21820, 2021. 2

[51] Yuejiang Liu, Riccardo Cadei, Jonas Schweizer, Sherwin Bahmani, and Alexandre Alahi. Towards robust and adaptive motion forecasting: A causal representation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17081–17092, 2022. 1

[52] Weibo Mao, Chenxin Xu, Qi Zhu, Siheng Chen, and Yanfeng Wang. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5517–5526, 2023. 1

[53] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14765–14775, 2022. 5

[54] M Jehanzeb Mirza, Inkyu Shin, Wei Lin, Andreas Schriebl, Kunyang Sun, Jaesung Choe, Mateusz Kozinski, Horst Possegger, In So Kweon, Kuk-Jin Yoon, et al. Mate: Masked autoencoders are online 3d test-time learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16709–16718, 2023. 2

[55] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David J Weiss, Benjamin Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*, 2022. 1

[56] 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. 2

[57] 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. 1

[58] Mozhgan Pourkeshavarz, Changhe Chen, and Amir Rasouli. Learn tarot with mentor: A meta-learned self-supervised approach for trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8384–8393, 2023. 2

[59] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019. 2

[60] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. Fjmp: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13745–13755, 2023. 1

[61] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020. 2

[62] Liushuai Shi, Le Wang, Chengjiang Long, Sanping Zhou, Wei Tang, Nanning Zheng, and Gang Hua. Representing multimodal behaviors with mean location for pedestrian trajectory prediction. *IEEE transactions on pattern analysis and machine intelligence*, 2023. 7

[63] Liushuai Shi, Le Wang, Sanping Zhou, and Gang Hua. Trajectory unified transformer for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9675–9684, 2023. 2

[64] Jianhua Sun, Yuxuan Li, Hao-Shu Fang, and Cewu Lu. Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13250–13259, 2021. 7

[65] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020. 2, 3

[66] Devavrat Tomar, Guillaume Vray, Behzad Bozorgtabar, and Jean-Philippe Thiran. Tesla: Test-time self-learning with automatic adversarial augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20341–20350, 2023. 2

[67] Li-Wu Tsao, Yan-Kai Wang, Hao-Siang Lin, Hong-Han Shuai, Lai-Kuan Wong, and Wen-Huang Cheng. Social-ssl: Self-supervised cross-sequence representation learning based on transformers for multi-agent trajectory prediction. In *European Conference on Computer Vision*, pages 234–250. Springer, 2022. 2

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[69] Chunnan Wang, Xiang Chen, Junzhe Wang, and Hongzhi Wang. Atpfl: Automatic trajectory prediction model design under federated learning framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6563–6572, 2022. 2

[70] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. 5

[71] Eason Wang, Henggang Cui, Sai Yalamanchi, Mohana Moorthy, and Nemanja Djuric. Improving movement predictions of traffic actors in bird's-eye view models using gans and differentiable trajectory rasterization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2340–2348, 2020. 2

[72] Jingke Wang, Tengju Ye, Ziqing Gu, and Junbo Chen. Ltp: Lane-based trajectory prediction for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17134–17142, 2022. 2

[73] 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, 2, 5

[74] Letian Wang, Yeping Hu, Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Changliu Liu. Transferable and adaptable driving behavior prediction. *arXiv preprint arXiv:2202.05140*, 2022. 2

[75] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 2

[76] Zhibo Wang, Jiayu Guo, Haiqiang Zhang, Ru Wan, Junping Zhang, and Jian Pu. Bridging the gap: Improving domain generalization in trajectory prediction. *IEEE Transactions on Intelligent Vehicles*, 2023. 1

[77] Chenfeng Xu, Tian Li, Chen Tang, Lingfeng Sun, Kurt Keutzer, Masayoshi Tomizuka, Alireza Fathi, and Wei Zhan. Pretram: Self-supervised pre-training via connecting trajectory and map. In *European Conference on Computer Vision*, pages 34–50. Springer, 2022. 2

[78] Chenxin Xu, Robby T Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1410–1420, 2023. 2

[79] Pei Xu, Jean-Bernard Hayet, and Ioannis Karamouzas. Socialvae: Human trajectory prediction using timewise latents. In *European Conference on Computer Vision*, pages 511–528. Springer, 2022. 2

[80] Yi Xu, Lichen Wang, Yizhou Wang, and Yun Fu. Adaptive trajectory prediction via transferable gnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6520–6531, 2022. 1, 2

[81] Yi Xu, Armin Bazarjani, Hyung-gun Chi, Chiho Choi, and Yun Fu. Uncovering the missing pattern: Unified framework towards trajectory imputation and prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9632–9643, 2023. 1

[82] Zhijie Yan, Pengfei Li, Zheng Fu, Shaocong Xu, Yongliang Shi, Xiaoxue Chen, Yuhang Zheng, Yang Li, Tianyu Liu, Chuxuan Li, et al. Int2: Interactive trajectory prediction at intersections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8536–8547, 2023. 1

[83] Tao Yang, Shenglong Zhou, Yuwang Wang, Yan Lu, and Nanning Zheng. Test-time batch normalization. *arXiv preprint arXiv:2205.10210*, 2022. 2

[84] Luyao Ye, Zikang Zhou, and Jianping Wang. Improving the generalizability of trajectory prediction models with frenét-based domain normalization. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11562–11568, 2023. 2

[85] Jiangbei Yue, Dinesh Manocha, and He Wang. Human trajectory prediction via neural social physics. In *European Conference on Computer Vision*, pages 376–394. Springer, 2022. 2

[86] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019. 5

[87] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, 35: 38629–38642, 2022. 2

[88] Xingxuan Zhang, Peng Cui, Renzhe Xu, Linjun Zhou, Yue He, and Zheyan Shen. Deep stable learning for out-of-distribution generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5372–5382, 2021. 1

[89] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023. 1

[90] Dekai Zhu, Guangyao Zhai, Yan Di, Fabian Manhardt, Hendrik Berkemeyer, Tuan Tran, Nassir Navab, Federico Tombari, and Benjamin Busam. Ipcc-tp: Utilizing incremental pearson correlation coefficient for joint multi-agent trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5507–5516, 2023. 1

[91] Yiyao Zhu, Di Luan, and Shaojie Shen. Biff: Bi-level future fusion with polyline-based coordinate for interactive trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8260–8271, 2023. 2