2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

MetaDAT: Generalizable Trajectory Prediction via Meta Pre-training and Data-Adaptive Test-Time Updating

Yuning Wang[1], Pu Zhang[2], Yuan He[2], Ke Wang[2], Jianru Xue[1]

***Abstract*— Existing trajectory prediction methods exhibit significant performance degradation under distribution shifts during test time. Although test-time training techniques have been explored to enable adaptation, current approaches rely on an offline pre-trained predictor that lacks online learning flexibility. Moreover, they depend on fixed online model updating rules that do not accommodate the specific characteristics of test data. To address these limitations, we first propose a meta-learning framework to directly optimize the predictor for fast and accurate online adaptation, which performs bi-level optimization on the performance of simulated test-time adaptation tasks during pre-training. Furthermore, at test time, we introduce a data-adaptive model updating mechanism that dynamically adjusts the predefined learning rates and updating frequencies based on online partial derivatives and hard sample selection. This mechanism enables the online learning rate to suit the test data, and focuses on informative hard samples to enhance efficiency. Experiments are conducted on various challenging cross-dataset distribution shift scenarios, including nuScenes, Lyft, and Waymo. Results demonstrate that our method achieves superior adaptation accuracy, surpassing state-of-the-art test-time training methods for trajectory prediction. Additionally, our method excels under suboptimal learning rates and high FPS demands, showcasing its robustness and practicality.**

I. Introduction

Trajectory prediction plays a crucial role in understanding environments and building world models, making it a cornerstone of autonomous driving [1], [2]. Mainstream research [3], [4], [5], [6] focuses on building strong data-driven predictors on pre-collected datasets, which is referred to as *offline training*. However, these predictors often struggle with distribution shifts in test data, such as changes in road structures [7], interaction patterns [8], and driving styles [9], leading to significant performance degradation. Such limitations pose critical safety risks and highlight the need for more robust solutions.

In this paper, we investigate *test-time training (TTT)* for trajectory prediction [10], [11], [12] to handle distribution shifts during testing. Our approach adapts a model pre-trained on *a source dataset* to an *unseen target domain* by online learning on the test-time data. Thanks to the auto-labeling nature of the trajectory prediction task, the observations at the current time work as labels for previous predictions. The training and evaluation protocol of TTT is illustrated in Fig. 1.

[1]Yuning Wang and Jianru Xue are with the State Key Laboratory of Human-Machine Hybrid Augmented Intelligence, IAIR, Xi'an Jiaotong University, China. Corresponding author: Jianru Xue. wangyn@stu.xjtu.edu.cn. jrxue@mails.xjtu.edu.cn.

[2]Pu Zhang, Yuan He, and Ke Wang are with KargoBot, China.

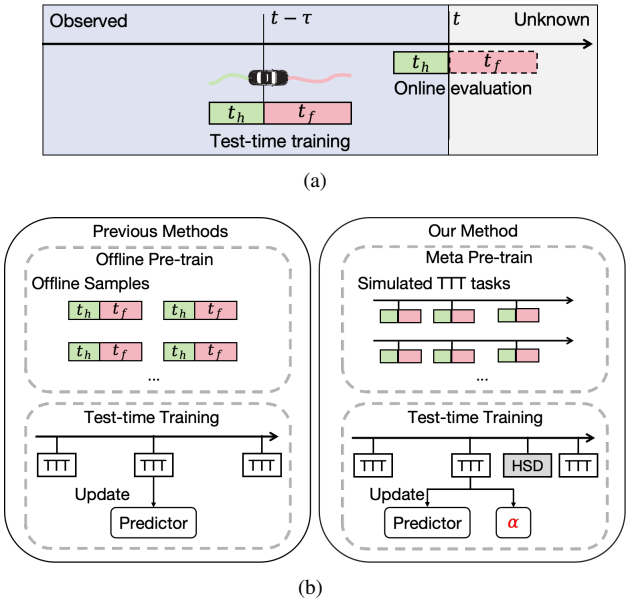

Fig. 1. (a) Illustration for the test-time training and online evaluation protocol for trajectory prediction. (b) Previous offline pre-training overlooks the offline-online misalignment between pre-training and adaptation. In contrast, our method aligns these two phases via meta pre-training (MP). Additionally, instead of using fixed updating rules, we propose data-adaptive test-time updating by dynamic optimization of the learning rate α (DLO) and additional hard-sample-driven updates (HSD).

Existing trajectory predictors demonstrate constrained online learning effectiveness, primarily due to two critical issues. **Firstly, existing offline pre-training objectives misalign with online adaptation**. Current pre-training objectives focus on offline prediction accuracy for in-distribution samples but overlook online adaptation capability. As a result, the predictor adapts slowly, and the pre-trained representations quickly deteriorate. **Secondly, current fixed online updating strategies cannot suit the varying test-time data.** An effective model updating rule should suit the test data characteristics. For example, the learning rate should be related to the magnitude of the distribution shift between the training and the unknown test data and be adaptively adjusted during learning. Also, efficient model updating should focus on the most representative samples for the distribution shift. Conventional methods, however, rely on rigid, predetermined learning rates and update frequencies and are unaware of the test data, severely limiting their performance and efficiency.

To tackle the aforementioned problem, we design a new TTT framework for trajectory prediction called MetaDAT: Meta pre-training and Data-Adaptive Test-time updating. Our framework introduces a bi-level optimization process in the pre-training stage based on meta-learning, to directly optimize

a flexible starting point for test-time updating. We align the offline training with online objectives by simulating test-time training tasks on the source dataset, and optimizing a meta-objective across those tasks that evaluates the online adapting performance. For the test-time updating process, we propose a unique learning rate optimization based on the online partial derivatives. Additionally, a hard-sample-driven method is proposed to focus the model updates on the most critical samples. Our contributions can be summarized as follows.

- We introduce a meta pre-training framework for test-time training that addresses the offline-online misalignment during pre-training, fully leveraging the potential of the pre-training stage and initializing a flexible predictor for subsequent adaptation.
- We propose a data-adaptive updating method for the test-time learning process, incorporating dynamic learning rates optimization and hard-sample-driven model updates, enabling effective and efficient adaptation.
- Experiments across widely used datasets and different configurations demonstrate that our method achieves state-of-the-art performance in both accuracy and efficiency. Our method can also make improvements under suboptimal learning rates and few-shot cases.

II. Related Work

A. Trajectory Prediction

Recently, data-driven trajectory predictors have achieved remarkable performance [5], [6], [13], [14] by utilizing large-scale pre-collected datasets [15], [16], [17], [18] and advanced techniques such as transformer architectures [5], [6], [19] and masked autoencoder (MAE) loss [5], [20]. Despite their high accuracy, research shows that the offline trained models are prone to test-time distribution shifts [8], [11], [10], [9]. Some work investigates transfer learning between distributions by anticipating the test domains [8], [7]. Those methods are unreliable when the domain shifts differently from anticipation. In contrast, our work focuses on test-time training for trajectory prediction, adapting to unknown domain shifts without prior assumptions.

B. Test-Time Training

Test-time training updates the model in an online manner on the newly observed data during testing. Recent TTT methods for image classification focus primarily on developing self-supervised losses to address the lack of labels during testing [21], [22], [23], [24], [25]. Trajectory prediction, however, has a unique auto-labeling nature, making TTT for this task a distinct problem with label supervision. Several TTT methods specifically designed for prediction have been developed. For instance, MEK [12] employs an extended Kalman filter as the online optimizer, while T4P [10] incorporates an MAE loss and actor-specific tokens. None of the previous methods considered the offline-online misalignment. The most relevant method is AML [11], which also utilizes meta-learning. However, AML only adapts the last Bayesian linear regression layer in the decoder, which limits the adaptation ability of deeper model representations. In contrast, we introduce a more universal meta pre-training framework to unleash the full model potential. Furthermore, we introduce a unique data-adaptive test-time updating technique.

C. Meta-Learning For Trajectory Prediction

Meta-learning focuses on making models learn how to learn for different tasks [26], [27], [28], [29], [30]. These methods have made significant progress in areas such as few-shot learning [26], [27], [28] and domain adaptation [31], [12], which is crucial for the generalizability of autonomous driving. Recently, some studies have explored meta learning for trajectory prediction. MENTOR [32] uses meta learning to perform multi-task learning, balancing auxiliary losses with the main prediction loss. MetaTraj [33] uses meta-learning to perform zero-shot transfer learning across scenes. None of those methods considered solving the offline-online objective misalignment for test-time training of the predictor.

III. Methodology

A. Problem Formulation

Trajectory Prediction. Trajectory prediction is a sequential prediction problem. At time step t, given the observation $\mathbf{X}_t = \{\mathcal{X}_t, \mathcal{M}_t\}$, where $\mathcal{X}_t = \mathbf{x}_{t-t_h:t}^{0:N}$ is the N agent trajectories over the past horizon t_h and $\mathcal{M}_t$ is the surrounding map information, the goal is to narrow the gap between the predictions $\hat{\mathbf{Y}}_t$ and the ground truth locations of agents $\mathbf{Y}_t = \mathbf{y}_{t:t+t_f}^{0:N}$ in future horizon t_f. For simplicity, we omit N in the following sections. In this work, we consider the situation where the test-time *target data* $D^T = \{\mathbf{X}^T, \mathbf{Y}^T\}$ does not share the same distribution as the training *source data* $D^S = \{\mathbf{X}^S, \mathbf{Y}^S\}$.

Test-time Training And Online Evaluation. We perform test-time training to adapt to D^T, and conduct online evaluations to assess adaptation performance, as illustrated in Fig. 1. We adopt the widely used online trajectory prediction setting [10], [11], [12], where the test data consists of multiple *scenes*, each containing *temporally ordered data samples*, and one sample is observed at each time step as time passes. Given a time interval τ, at time t we can get the ground truth future $\mathbf{Y}_{t-\tau} = \mathbf{x}_{t-\tau:t-\tau+t_f}$ for previous prediction at time $t-\tau$, thus we can update the predictor with supervision data $\{\mathbf{X}_{t-\tau}, \mathbf{Y}_{t-\tau}\}$. The online evaluation at time t is performed on the prediction $\mathbf{Y}_t$ for current $\mathbf{X}_t$.

B. Overview

We propose a test-time training framework for trajectory prediction called MetaDAT. The overview of MetaDAT is illustrated in Fig. 2. MetaDAT consists of a *meta pre-training* phase (Fig. 2 (c)) and a *data-adaptive test-time updating* phase (Fig. 2 (d)). During model pre-training, pseudo TTT tasks are simulated through temporal scene splitting. Meta-learning is employed on these tasks to derive a flexible model initialization θ^* optimized for subsequent updating. During test-time updating, we propose online dynamic optimization of the learning rate α and perform hard-sample-driven model updates, ensuring that the online learning process suits the test-time data and concentrates on the critical samples.

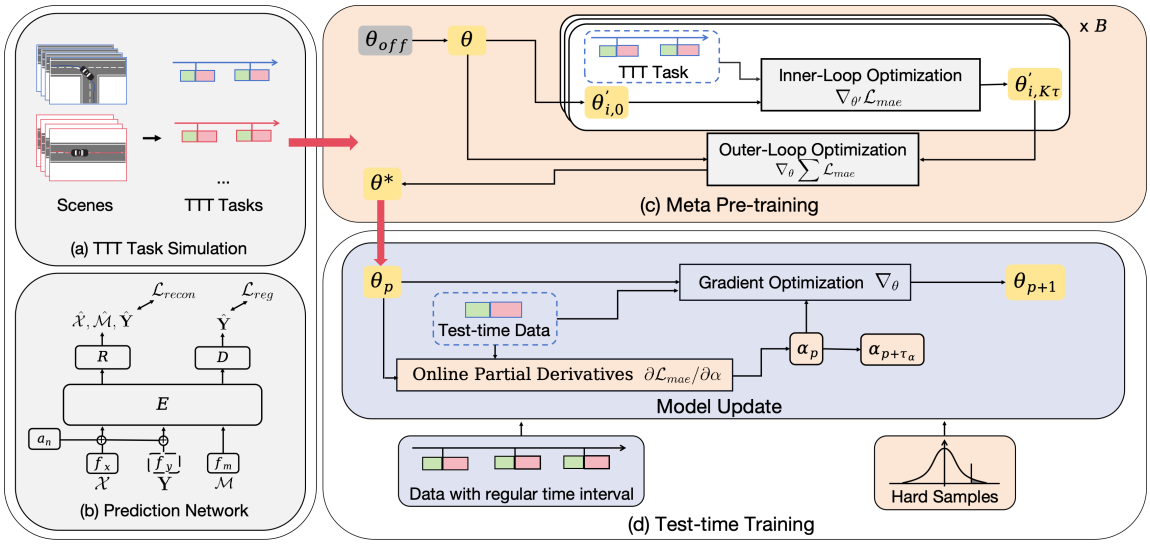

Fig. 2. Overview of our MetaDAT framework. MetaDAT first performs meta pre-training across the simulated test-time training tasks on the source dataset, then the pre-trained model goes through data-adaptive test-time training on the target dataset under distribution shifts. Newly proposed modules in our framework are highlighted in orange, while the TTT baseline is in purple.

C. Predictor Model

For our prediction network in Fig. 2.(b), we adopt the SOTA predictor ForecastMAE [5] as the backbone, the same as previous work T4P [10] to enable fair comparison. ForecastMAE consists of embedding layers $\{f_x, f_y, f_m\}$, an encoder E, a decoder D, and an MAE reconstruction branch R. Note that the future input $\mathbf{Y}$ is only for training.

Pre-training Stage. We adopt the modification in T4P [10]: instead of using a two-stage training [5], the model parameter $\theta = \{f, E, D, R\}$ is optimized jointly on the MAE loss $\mathcal{L}_{mae}$, which consists of the regression loss $\mathcal{L}_{reg}$ and the reconstruction loss $\mathcal{L}_{recon}$:

$$\min_{\theta} \mathbb{E}_{D^s} \mathcal{L}_{mae} = \mathcal{L}_{reg}(\mathbf{X}, \mathbf{Y}) + \mathcal{L}_{recon}(\mathbf{X}, \mathbf{Y}) \quad (1)$$

Test-time Training. At time step t, we update the model by minimizing the MAE loss at the previous time step $t-\tau$ given the TTT time interval τ.

$$\min_{\theta} \mathcal{L}_{mae}^{t-\tau} = \mathcal{L}_{reg}(\mathbf{X}_{t-\tau}, \mathbf{Y}_{t-\tau}) + \mathcal{L}_{recon}(\mathbf{X}_{t-\tau}, \mathbf{Y}_{t-\tau}) \quad (2)$$

We also adopt the actor-specific tokens proposed by T4P [10] to learn actor-wise habits. In the TTT stage, the actor-specific tokens a_n are added to the embedding for each actor n to obtain the actor-wise feature h_x, h_y, then h_x, h_y are fed into the encoder E.

$$h_x, h_y = f_x(\mathbf{X}) + a_n, f_y(\mathbf{Y}) + a_n \quad (3)$$

D. Meta Pre-training.

We propose a meta pre-training (MP) framework to directly optimize the pre-trained model θ for test-time training on D^s, solving the offline-online misalignment.

TTT Task Simulation. To optimize the model for test-time adaptation, we first partition D^s into separate sub-domains and simulate online test-time training tasks. For trajectory prediction, prior research has shown that different driving scenes naturally represent consistent sub-domains [8], [10], [33], as each scene comprises unique agent behavioral patterns and distinct road structures. Thus, we regard driving scenes as sub-domains and simulate TTT tasks on them. Specifically, we divide the source dataset into individual driving scenes, and organize the samples within each scene s of length t_s in temporal order to create the online trajectory sequence $\mathbf{S} = \{\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{t_s}\}$. This process generates an online sequence set $\mathcal{S} = \{\mathbf{S}_0, \mathbf{S}_1, ...\}$ from all the scenes in D^s, and the TTT task set $\mathcal{T} = \{\mathbf{T}_0, \mathbf{T}_1, ...\}$ is defined by online learning on $\mathcal{S}$.

Meta-Learning. Given the initial model parameter θ and the TTT task set $\mathcal{T}$, we perform a bi-level optimization [26]: in the inner loop, the model θ undergoes simulated online test-time training to yield θ'; in the outer loop, a meta-objective evaluates the adaptation performance and optimizes the initial parameters θ. This process is illustrated in Algorithm 1. We

Algorithm 1 Meta Pre-training

Require: inner learning rate α_{in}, meta-learning rate β, initial model weights $\theta \leftarrow \theta_{off}$, TTT task set $\mathcal{T}$, meta-learning batch size B, inner-loop step K, time interval τ.

while not converge **do**
  sample TTT task batch $\{\mathbf{T}_0, ...\mathbf{T}_B\} \sim \mathcal{T}$
  for each $\mathbf{S}_i$ **do**
    initialize model parameter $\theta'_{i,0} = \theta$
    for $t = \tau, 2\tau..., K\tau$ **do**
      Compute adapted parameters with gradient descent: $\theta'_{i,\tau} = \theta'_{i,t-\tau} - \alpha_{in} \nabla_{\theta'_{i,t-\tau}} \mathcal{L}_{mae}^{i,t-\tau}(\theta'_{i,t-\tau})$
    end for
  end for
  Update meta-parameters: $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_i \mathcal{L}_{mae}^{i,K\tau}$
end while
return optimized model weights θ^*

first sample a task batch $\{\mathbf{T}_0, ...\mathbf{T}_B\}$ with batch size B. For a single online TTT task $\mathbf{T}_i$, we initialize task-specific model parameters $\theta'_{i,0} = \theta$ at $t = 0$, then θ'_i is optimized by performing K step test-time training in the inner loop. For each training step $t = \tau, 2\tau..., K\tau$,

$$\theta'_{i,\tau} = \theta'_{i,t-\tau} - \alpha_{in} \nabla_{\theta'_{i,t-\tau}} \mathcal{L}^{i,t-\tau}_{mae}(\theta'_{i,t-\tau}). \tag{4}$$

The adaptation performance is evaluated by the prediction loss $\mathcal{L}^{i,K\tau}_{mae}$ after the last adaptation step. After B inner loops, we perform meta optimization across the B evaluation loss $\mathcal{L}^{i,K\tau}_{mae}$s with respect to the initial parameter θ in the outer loop. The optimization on θ is as follow:

$$\theta = \theta - \beta \nabla_\theta \sum_i \mathcal{L}^{i,K\tau}_{mae} \tag{5}$$

This equation contains second-order gradients with Hessian-vector products, which is computation consuming. In practice, we use the first-order-approximation [34]s.

Model Initialization. The inner step of meta-learning makes the training process very time-consuming. Also, prior work [28] reveals that performing meta-learning over an offline pre-trained model can result in a model with better generalization. Therefore, instead of training from scratch, we initialize the model with offline pre-trained parameters θ_{off}, $\theta = \theta_{off}$ before meta-learning.

E. Data Adaptive Test-time Updating.

In addition to a good initialization, the test-time model updating rule is also critical to the adaptation performance. Conventional online updating methods adopt fixed learning rates and update frequencies, which fails to account for the unknown characteristics of test data. Recognizing this issue, we propose a data-adaptive mechanism for test-time updating.

1) Dynamic Learning Rate Optimization (DLO): We employ online partial derivatives to dynamically optimize the learning rate according to the observed performance. For simplicity, we consider stochastic gradient descent for parameter θ_p at the pth TTT step with learning rate α,

$$\theta_p = \theta_{p-1} - \alpha \nabla_\theta \mathcal{L}_{mae}(\theta_{p-1}), \tag{6}$$

It is straightforward to extend to more complex optimizers such as AdamW [35].

Inspired by [36], taking the assumption that the optimal α does not change much between two consecutive updates, we can use the partial derivative $\partial \mathcal{L}_{mae}(\theta_{p-1})/\partial\alpha$ on step $p-1$ to optimize α on step p. Noting $\theta_{p-1} = \theta_{p-2} - \alpha \nabla_\theta \mathcal{L}_{mae}(\theta_{p-2})$ and applying the chain rule, we can get:

$$\begin{aligned} \frac{\partial \mathcal{L}_{mae}(\theta_{p-1})}{\partial \alpha} &= \nabla_\theta \mathcal{L}_{mae}(\theta_{p-1}) \cdot \\ &\quad \frac{\partial(\theta_{p-2} - \alpha \nabla_\theta \mathcal{L}_{mae}(\theta_{p-2}))}{\partial \alpha} \\ &= \nabla_\theta \mathcal{L}_{mae}(\theta_{p-1}) \cdot (-\nabla_\theta \mathcal{L}_{mae}(\theta_{p-2})) \end{aligned} \tag{7}$$

And then α is updated as follows:

$$\begin{aligned} \alpha_p &= \alpha_{p-1} - \gamma \partial \mathcal{L}_{mae}(\theta_{p-1})/\partial\alpha \\ &= \alpha_{p-1} + \gamma \nabla_\theta \mathcal{L}_{mae}(\theta_{p-1}) \cdot \nabla_\theta \mathcal{L}_{mae}(\theta_{p-2}), \end{aligned} \tag{8}$$

where γ is the learning rate for α.

In practice, we update the learning rate by averaging over a longer interval τ_α to stabilize the training process:

$$\alpha_p = \alpha_{p-\tau_\alpha} + \gamma \nabla_\theta \mathcal{L}_{mae}(\theta_{p-1}) \cdot \frac{1}{\tau_\alpha} \sum_{j=p-\tau_\alpha-1}^{p-2} \nabla_\theta \mathcal{L}_{mae}(\theta_j), \tag{9}$$

We adopt different αs for each network layer, with the same initial learning rate α_{init}. This can better liberate the network flexibility of the adaptive process without the additional need for manual hyperparameter tuning.

2) Hard-sample-driven Model Updates (HSD): Autonomous driving data exhibits a long-tail distribution, where a small fraction of challenging data—such as scenarios involving intensive interactions or heavy reliance on road maps—is particularly critical and demands sophisticated modeling [37], [38]. These data are most susceptible to performance degradation under distribution shift and best represent the information that needs to be learned in the current domain. We identify those critical hard samples by comparing the prediction error e with a running error mean m and a running error standard deviation σ as follows, and perform additional updates on them:

$$e > m + k\sigma. \tag{10}$$

Equation 10 restricts the updates to only a small number of samples. Therefore, we can utilize hard-sample-driven updates without sacrificing efficiency. Learning rate optimization is not implemented on these hard samples.

IV. Experiments

A. Datasets

We conduct experiments on well-known trajectory datasets: Waymo [15],nuScenes [16], Lyft [18], to construct various cross-dataset distribution shifts. We make sure that scene lengths in the source dataset are enough to perform a single-step test-time update for meta pre-training. Trajdata [39] is used as the dataset interface. Two kinds of widely used configurations are adopted: long-term prediction and short-term prediction. For short-term prediction, we watch 1 second and predict 3 seconds with a time interval of 0.1. For long-term prediction, we watch 2 seconds and predict 6 seconds with a time interval of 0.5. We evaluate the best-of-6 metrics [40] for multi-modal prediction: $mADE_6$ and $mFDE_6$. We also evaluate $mADE_1$ and MR_6 for comprehensive analysis.

B. Baselines.

We compare with widely used unsupervised/supervised test-time-training methods: DUA [41] and TENT [42] with supervision, as-well-as state-of-the-art TTT methods specifically for trajectory prediction: MEK [12], AML [11], and T4P [10]. All methods use the same network architecture as ours. We also add a **non-TTT baseline**: Joint-training, which means only pre-training the model on the source dataset using the joint MAE loss $\mathcal{L}_{mae}$ and performing no test-time update.

TABLE I

PERFORMANCE ON VARIOUS DISTRIBUTION SHIFTS. THE PREDICTOR IS TRAINED ON THE SOURCE DATASET, AND TEST-TIME TRAINED AND ONLINE EVALUATED ON THE TARGET DATASET (SOURCE →TARGET). * MEANS REPRODUCED RESULTS BY OURSELVES.

$mADE_6$ /$mFDE_6$	Short-term prediction (1/3/0.1)				long-term prediction (2/6/0.5)		
	Lyft→nuS	nuS→Way	Way→nuS	Mean	nuS→Lyft	Lyft→nuS	Mean
Joint Training	0.506/1.102	0.472/1.125	0.374/0.718	0.450/0.982	1.108/2.597	1.398/3.109	1.253/2.853
DUA	0.474/1.118	0.516/1.294	0.396/0.781	0.462/1.064	1.365/3.257	1.585/3.607	1.475/3.432
TENT(w/ sup)	0.462/1.009	0.448/1.071	0.358/0.695	0.423/0.925	1.068/2.514	1.396/3.102	1.232/2.808
MEK	0.508/1.806	0.405/1.061	0.373/0.713	0.429/1.192	1.006/2.369	1.426/3.119	1.216/2.744
AML	0.454/1.954	0.764/1.791	0.483/0.962	0.567/1.569	1.462/2.573	1.698/3.367	1.580/2.970
T4P	0.357/0.770	0.336/0.807	0.323/0.656	0.339/0.744	0.776/1.820	1.254/2.802	1.014/2.311
T4P*	0.408/0.847	0.343/0.792	0.284/0.585	0.345/0.741	0.711/1.578	1.260/2.742	0.985/2.160
MetaDAT(Ours)	**0.332/0.683**	**0.305/0.712**	**0.266/0.548**	**0.301/0.648**	**0.648/1.472**	**1.177/2.551**	**0.912/2.011**

TABLE III

COMPARISON ON MORE MULTI-MODAL PREDICTION PERFORMANCE.

$mADE_1$ /MR_6	Short-term prediction		
	Lyft→nuS	nuS→Way	Way→nuS
T4P	1.050/0.088	0.805/0.091	0.911/0.037
MetaDAT	**0.856/0.060**	**0.708/0.078**	**0.785/0.026**

TABLE IV

ABLATIONS ON DIFFERENT MODULES OF METADAT.

Components			Short-term	Long-term	Mean
MP	DLO	HSD	Lyft→nuS	nuS→Lyft	
			0.408/0.847	0.711/1.578	0.560/1.213
✓			0.355/0.734	0.672/1.491	0.514/1.112
	✓		0.376/0.776	0.684/1.538	0.530/1.157
		✓	0.400/0.836	0.707/1.552	0.554/1.194
✓	✓		0.347/0.702	0.650/**1.468**	0.498/1.085
✓	✓	✓	**0.332/0.683**	**0.648**/1.472	**0.490/1.077**

C. Implement Details.

We use the same model structure as ForecastMAE [5], [10]. For offline pre-training, we adopt the same training schedule as T4P [10]. The offline pre-trained model works as model initialization for meta pre-training. For meta pre-training, we train for 8 epochs with a meta batch size $B = 4$ using the AdamW [35] optimizer with weight decay of 0.001. The inner-loop step K is set as 4. The learning rate β for meta pre-training is set as 5e-4 initially and is gradually decayed by a cosine scheduler to 1e-6. The inner learning rate α_{in} is the same as the online learning rate α_{init}. For the test-time training process, we select the initial learning rate α_{init} as 0.001 for nuS →Way short-term and all the long-term experiments, and 0.01 for all other experiments. The AdamW optimizer [35] is adopted for test-time training. We select the TTT time interval $\tau = t_f$ to allow the past GT future to contain the full prediction horizon. γ is set as 1e-4, and the learning rate update interval τ_α is 8. k is set as 3. In Sec. IV-E, the FPS results are calculated on a single NVIDIA-L4 GPU.

D. Results And Comparisons.

Quantitative Results. Table I summarizes our experimental results under various cross-dataset distribution shifts. All the results are averaged over 3 independent runs. Our method stably outperforms all comparison methods across various distribution shifts and different prediction configurations, showing the superiority and generalization ability of MetaDAT. Specifically, we outperform the second-best TTT method T4P [10] by 12.7% on $mADE_6$ and 12.5% on $mFDE_6$ under short-term prediction configuration, thanks to our meta pre-training to get a flexible model initialization, and our effective and efficient data-adaptive updating. Moreover, our method outperforms another meta-learning-based competitor AML [11] significantly. This highlights that our method has the flexibility to pre-train and finetune deeper model representations. We also compare with the second-best T4P on more metrics in Tab. III, to better show our superior multi-modal prediction performance.

Qualitative Results. In Figure 3 we visualize the qualitative prediction results after test-time training on various driving scenarios, such as going straight, turning, and crossing intersections. We compare against two methods: Joint-training which performs no adaptation, and the previous SOTA TTT method T4P. Our method shows stable superior adaptation performance, surpassing both methods across various datasets. This demonstrates the effectiveness of the strategies we proposed in both pre-training and online training. Furthermore, qualitative comparison on the multi-modal prediction results in Fig. 4 shows that our method can enhance both horizontal and vertical prediction modality diversities.

E. Ablation And Discussions

Ablations. In Tab. IV we do ablations on different components of MetaDAT. The table illustrates that all three proposed modules: meta pre-training, dynamic learning rate optimization, and hard-sample-driven updates can lead to performance improvements when working individually. Combining all three modules leads to further improvements. This illustrates the complementarity of those modules, as they simultaneously focus on two different aspects: pre-training a powerful model initialization, and more adaptive test-time training by adjusting learning rate and updating frequency.

Robustness To Different Learning Rate. Robustness is a critical factor in online learning for trajectory prediction, as it significantly impacts the safety of real-time systems. We studied the robustness of the test-time training algorithms to the initial online learning rates $\alpha(\alpha_{init})$ in Tab. V. We can see that α largely influences the performance of T4P. Our proposed dynamic learning rate optimization can adaptively

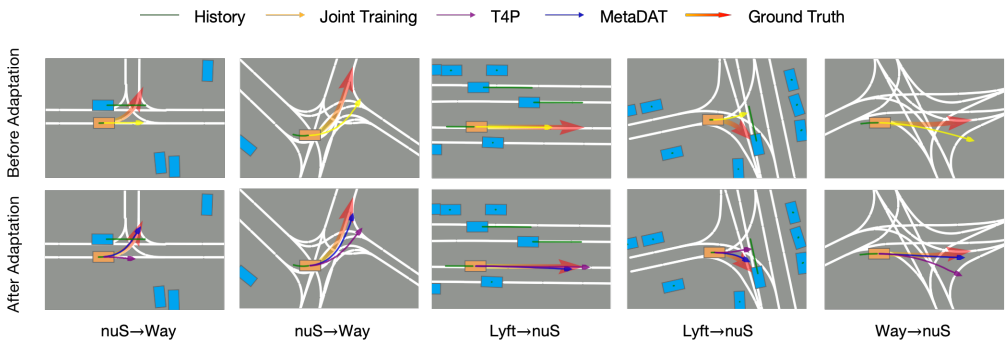

Fig. 3. Qualitative visualizations on test-time training results. The top row shows the prediction results without adaptation, while the bottom row indicates the adaptation results using test-time training. We only visualize the best modality for multi-modal predictions.

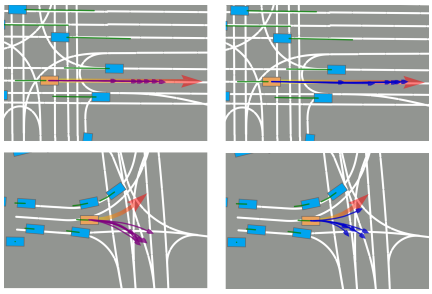

Fig. 4. Qualitative visualizations on multi-modal prediction results for TTT methods. Left: T4P. Right: MetaDAT.

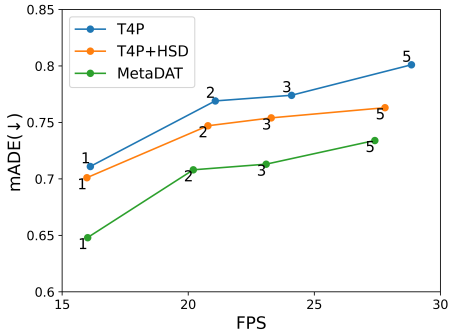

Fig. 5. Prediction accuracy and execution FPS on nuS→Lyft long-term experiments. The number next to each data point indicates the updating frequency. We plot results for frequencies of 1,2,3,5. MetaDAT demonstrates better prediction performance at the same FPS.

TABLE V

ROBUSTNESS TO LEARNING RATE $\alpha(\alpha_{init})$ ON NUS→WAY SHORT-TERM $(mADE_6/mFDE_6)$.

	α=0.01	α=0.001	α=0.0001
T4P	0.518/1.097	0.343/0.792	0.393/0.967
T4P+DLO	0.452/0.957	0.337/0.762	0.350/0.792
MetaDAT	**0.407/0.887**	**0.305/0.712**	**0.341/0.760**

adjust α to suit the test data, therefore the performance under bad initial α is improved (T4P+DLO). Our full MetaDAT makes more progress, which might be because the meta pre-training can get a robust model initialization.

Efficiency. Efficiency is another crucial factor for online learning. In Fig. 5 we evaluate the frame per second (FPS) along with accuracy ($mADE_6$) of MetaDAT against the second-best method T4P. Both methods can improve FPS by adjusting the update frequency: a frequency of 1 indicates updating at every opportunity, making $\tau = 12$, while 2 means $\tau = 24$. The HSD updating frequency is always 1. MetaDAT can improve the performances with little additional latency introduced, thanks to the hard-sample-driven updates.

Few-shot Adaptation Results. In Tab. VI we analyze the prediction performance with different sample amounts for test-time adaptation. The table clearly demonstrates that MetaDAT consistently outperforms T4P across various data amounts. Notably, MetaDAT achieves impressive performance even with a small number of samples, such as 2000, highlighting its few-shot adaptation capability.

TABLE VI

FEW-SHOT ADAPTATION PERFORMANCE $(mADE_6/mFDE_6)$ ON NUS→WAY SHORT-TERM.

Sample	2000	5000	10000
T4P	0.350/0.849	0.348/0.801	0.343/0.792
MetaDAT	**0.327/0.743**	**0.317/0.731**	**0.305/0.712**

Limitations And Future Work. Our test-time training approach relies on accurate online detection and tracking to obtain accurate observed trajectories for training. However, in practice, noisy trajectories from imperfect perception can degrade model performance [43], [44]. Thus, developing robust online learning strategies remains a key challenge.

V. CONCLUSION

We propose a Meta pre-training and Data Adaptive Test-time updating framework for trajectory predictors under distribution shift. Both quantitative and qualitative results demonstrate that our method achieves SOTA adaptation performance on various cross-dataset shifts and prediction configurations. Additionally, our approach enhances adaptation performance under few-shot conditions, suboptimal learning rates, and high-efficiency requirements, making it well-suited for real-world applications.

ACKNOWLEDGEMENT

This work is supported by the National Key Research and Development Program of China (Grant No. 2024YFE0210700)

REFERENCES

[1] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 4363–4369.

[2] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[3] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "Hivt: Hierarchical vector transformer for multi-agent motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8823–8833.

[4] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6531–6543, 2022.

[5] J. Cheng, X. Mei, and M. Liu, "Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8679–8689.

[6] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-centric trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 863–17 873.

[7] L. Ye, Z. Zhou, and J. Wang, "Improving the generalizability of trajectory prediction models with frenet-based domain normalization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 562–11 568.

[8] Y. Xu, L. Wang, Y. Wang, and Y. Fu, "Adaptive trajectory prediction via transferable gnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 6520–6531.

[9] A. B. Vasudevan, N. Peri, J. Schneider, and D. Ramanan, "Planning with adaptive world models for autonomous driving," *arXiv preprint arXiv:2406.10714*, 2024.

[10] D. Park, J. Jeong, S.-H. Yoon, J. Jeong, and K.-J. Yoon, "T4p: Test-time training of trajectory prediction via masked autoencoder and actor-specific token memory," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 065–15 076.

[11] B. Ivanovic, J. Harrison, and M. Pavone, "Expanding the deployment envelope of behavior prediction via adaptive meta-learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7786–7793.

[12] L. Wang, Y. Hu, and C. Liu, "Online adaptation of neural network models by modified extended kalman filter for customizable and transferable driving behavior prediction," *arXiv preprint arXiv:2112.06129*, 2021.

[13] Y. Zhou, H. Shao, L. Wang, S. L. Waslander, H. Li, and Y. Liu, "Smartrefine: A scenario-adaptive refinement framework for efficient motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 281–15 290.

[14] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.

[15] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[17] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. 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.

[18] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," in *Conference on Robot Learning*. PMLR, 2021, pp. 409–418.

[19] D. Park, H. Ryu, Y. Yang, J. Cho, J. Kim, and K.-J. Yoon, "Leveraging future relationship reasoning for vehicle trajectory prediction," *arXiv preprint arXiv:2305.14715*, 2023.

[20] H. Chen, J. Wang, K. Shao, F. Liu, J. Hao, C. Guan, G. Chen, and P.-A. Heng, "Traj-mae: Masked autoencoders for trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8351–8362.

[21] Y. Liu, P. Kothari, B. Van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "Ttt++: When does self-supervised test-time training fail or thrive?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 808–21 820, 2021.

[22] Y. Gandelsman, Y. Sun, X. Chen, and A. Efros, "Test-time training with masked autoencoders," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29 374–29 385, 2022.

[23] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International conference on machine learning*. PMLR, 2020, pp. 9229–9248.

[24] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 295–305.

[25] M. J. Mirza, I. Shin, W. Lin, A. Schriebl, K. Sun, J. Choe, M. Kozinski, H. Possegger, I. S. Kweon, K.-J. Yoon, *et al.*, "Mate: Masked autoencoders are online 3d test-time learners," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 16 709–16 718.

[26] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[27] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 403–412.

[28] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang, "Meta-baseline: Exploring simple meta-learning for few-shot learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9062–9071.

[29] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[30] K. Javed and M. White, "Meta-learning representations for continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[31] D. Li and T. Hospedales, "Online meta-learning for multi-source and semi-supervised domain adaptation," in *European Conference on Computer Vision*. Springer, 2020, pp. 382–403.

[32] M. Pourkeshavarz, C. Chen, and A. 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*, 2023, pp. 8384–8393.

[33] X. Shi, H. Zhang, W. Yuan, and R. Shibasaki, "Metatraj: meta-learning for cross-scene cross-object trajectory prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[34] A. Nichol, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[35] I. Loshchilov, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[36] A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood, "Online learning rate adaptation with hypergradient descent," *arXiv preprint arXiv:1703.04782*, 2017.

[37] O. Makansi, Ö. Çiçek, Y. Marrakchi, and T. Brox, "On exposing the challenging long tail in future prediction of traffic actors," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 147–13 157.

[38] Y. Wang, P. Zhang, L. Bai, and J. Xue, "Fend: A future enhanced distribution-aware contrastive learning framework for long-tail trajectory prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 1400–1409.

[39] B. Ivanovic, G. Song, I. Gilitschenski, and M. Pavone, "trajdata: A unified interface to multiple human trajectory datasets," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[40] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8748–8757.

[41] M. J. Mirza, J. Micorek, H. Possegger, and H. 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*, 2022, pp. 14 765–14 775.

[42] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Ful ly test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020.

[43] D. Park, J. Jeong, and K.-J. Yoon, "Improving transferability for cross-domain trajectory prediction via neural stochastic differential equation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 9, 2024, pp. 10 145–10 154.

[44] B. Ivanovic, Y. Lin, S. Shrivastava, P. Chakravarty, and M. Pavone, "Propagating state uncertainty through trajectory forecasting," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2351–2358.