Expanding the Deployment Envelope of Behavior Prediction via Adaptive Meta-Learning

Boris Ivanovic[1] James Harrison[2] Marco Pavone[3]

***Abstract*— Learning-based behavior prediction methods are increasingly being deployed in real-world autonomous systems, e.g., in fleets of self-driving vehicles, which are beginning to commercially operate in major cities across the world. Despite their advancements, however, the vast majority of prediction systems are specialized to a set of well-explored geographic regions or operational design domains, complicating deployment to additional cities, countries, or continents. Towards this end, we present a novel method for efficiently adapting behavior prediction models to new environments. Our approach leverages recent advances in meta-learning, specifically Bayesian regression, to augment existing behavior prediction models with an adaptive layer that enables efficient domain transfer via offline fine-tuning, online adaptation, or both. Experiments across multiple real-world datasets demonstrate that our method can efficiently adapt to a variety of unseen environments.**

I. Introduction

Learning-based behavior prediction methods are becoming a staple of the modern robotic autonomy stack, with nearly every major autonomous vehicle organization incorporating state-of-the-art behavior prediction models in their software stacks [1], [2], [3], [4], [5], [6], [7]. In order to deploy such systems safely and reliably alongside humans, organizations extensively train and test models in their specific operational domains. While this improves system accuracy and safety within the targeted domain, it does so at the cost of generalization. Specifically, deploying autonomous vehicles to different cities or countries remains challenging due to their different social interaction standards, laws, agent types, and road geometries (Fig. 1).

Currently, autonomous vehicles expand their operational domain through an extensive process of collecting and annotating large amounts of data from target locations [8]. However, manually annotating data is labor-intensive and expensive, and would be prohibitive for a goal of worldwide deployment. To enable widespread deployment without continual data collection and annotation, self-driving vehicles will need to be able to adapt to their surroundings, either from data observed online or from a small amount of already-annotated data.

Towards this end, in this work we present a method for adaptive trajectory forecasting that augments existing prediction models and enables them to efficiently adapt to new environments. Our approach employs meta-learning [9], specifically we extend the Bayesian regression approach of ALPaCA [10], to replace the last layer of a neural network with an adaptive Bayesian linear regression layer whose posterior distribution can be updated efficiently from newly-observed datapoints. As shown in Section VI, our method yields significant error reductions in the face of domain shifts compared to naïve transfer and adapts more quickly and efficiently than gradient descent-based finetuning.

Fig. 1. Deploying autonomous systems to a wide variety of diverse locations remains challenging, complicated by (clockwise from top-left) differing social interaction standards (e.g., traffic density in India), environment geometries (e.g., road/bike lane intersections in Ireland), agent types (e.g., Tuk-tuks in Thailand), and laws (e.g., forming a "Rettungsgasse" during heavy traffic in Germany), among other factors.

Contributions. Our key contributions are threefold. First, we present a trajectory forecasting algorithm capable of online updating and effective transfer through an adaptive last layer. We combine ALPaCA-based [10] last layer adaptation with recurrent models, and show that they work synergistically to enable both broad generalization and efficient transfer (Section V). Second, we present a novel trajectory forecasting algorithm based on Trajectron++ [11] that naturally pairs with the ALPaCA adaptive last layer. In particular, we introduce a within-episode aleatoric uncertainty prediction scheme that enables state-dependent multimodality and modulates last layer adaptation (Section V-B). Finally, we show experimentally that this architecture broadly extends the deployment envelope of trajectory forecasting algorithms, enabling efficient transfer to problem settings substantially different from those on which they were trained (Section VI).

II. Related Work

A. Adaptive Behavior Prediction

The field of behavior prediction has grown significantly in recent years, with many works focusing on improving model accuracy, incorporating social factors (e.g., inter-agent interactions), accounting for multimodality, and leveraging advancements in deep learning methodology [12]. Current

[1]Boris Ivanovic is with NVIDIA Research. bivanovic@nvidia.com

[2]James Harrison is with Google Research, Brain Team. jamesharrison@google.com

[3]Marco Pavone is with the Department of Aeronautics and Astronautics, Stanford University, and with NVIDIA Research. {pavone@stanford.edu,mpavone@nvidia.com}

state-of-the-art approaches particularly reflect this, employing graph-structured Recurrent Neural Networks (RNNs) with multimodality modeled by Conditional Variational Autoencoders (CVAEs) [13] (e.g., [11]) or Generative Adversarial Nets [14] (e.g., [15]), Transformers [16] (e.g., [17], [18]), and deep convolutional networks (popular in end-to-end approaches, e.g., [19], and for real-time use [20]). While there have been many advances, such works have only been evaluated in the same environments as they were trained.

At the same time, there have been a plethora of datasets released, with thousands of hours of data now publicly available. Accordingly, there have recently been studies into the cross-domain generalization of behavior prediction methods [21], as well as works tackling the problem of domain adaptation for trajectory forecasting. Broadly, adaptive behavior prediction methods can be categorized into the following three groups:

Memory. Memory-based approaches generally tackle domain shifts by first computing past and future trajectory embeddings (e.g., with recurrent neural networks) and then leveraging an associative external memory to store and retrieve the embeddings at test-time [22], [23]. Trajectory prediction is performed by decoding future embeddings (stored in memory) conditioned on the observed past and any context. Using memory also allows for continuous improvement online by storing novel trajectories. One key drawback of memory-based approaches, especially for online adaptation, is that constantly-growing extra memory may not be feasible for robotic platforms which have fixed memory and compute limits. In contrast, our method does not require any extra memory.

Architecture. Architectural methods generally employ neural networks whose intermediate representations or structures are generalizable and can transfer to different domains. Two recent examples include the Transferable Graph Neural Network (T-GNN) [24] and the Hierarchical, Adaptable, and Transferable Network (HATN) [25]. At a high level, both works propose a hierarchical structure, where one component learns domain-invariant trajectory features and another learns low-level, transferable features. A key drawback of such approaches is that they either require labeled data from the target domain during training-time [24] or are not inherently online-adaptive [25].

Least Squares. Methods in this category typically employ recursive least squares to adapt models to data observed online. Similar to our approach, RLS-PAA [26], [27] updates the last layer of a trained neural network via iterative least squares. However, since RLS-PAA is not performed during training, it is generally only useful for adapting to local agent behaviors within the same dataset (rather than for domain transfer). This is also reflected in RLS-PAA's use and performance in recent literature, e.g., [28], [29], [30] only adapt to local dynamics and agent behaviors within the same dataset. Further, while HATN [25] tackles cross-domain transfer, its ablation study shows that incorporating RLS-PAA provides no benefit. In contrast, our method back-propagates *through* the least squares updates during training, yielding a calibrated prior model that can be efficiently adapted to different domains.

B. Few-Shot Learning by Meta-Learning

In few-shot learning, a learner must learn to predict from a small amount of data [31]. Meta-learning exists as a particular approach to few-shot learning, in which transfer to a new domain is enabled by "learning to learn" across a set of training tasks [32], [33], [34], [35], [36], [37], [38], [39]. By learning update rules on many tasks, an agent may learn to learn effectively, and thus efficiently, in a new task.

While there are several ways to design meta-learning algorithms, two are of note for this work. First, *black-box* meta-learning exploits sequence-processing neural network models such as recurrent networks [35], [39], [40]. While expressive, these models have no particular inductive bias toward learning, and thus they are practically inefficient and require huge amounts of training data. To address this issue, *optimization-based* meta-learners [32], [10], [41], [37], which directly leverage optimization in the inner learning loop, have been a major research focus. The model that we develop in this paper can be viewed as a combined black-box and optimization meta-learner. In particular, it was previously shown that recurrence alone is not sufficient for effective transfer. We address this in this work by showing that recurrence-based adaptation can be effectively paired with optimization-based meta-learning.

III. Background: Adaptive Meta-Learning

Our framework leverages an *adaptive* formulation of meta-learning. In contrast to the more common episodic meta-learning (e.g., MAML [32]), the adaptive formulation allows tasks to vary over time within an episode, similar to meta-learning for continual learning [42]. In particular, our approach builds on an extension to ALPaCA [10] developed in [43]. Instead of Bayesian linear regression (as in ALPaCA), adaptation is done via Kalman filtering on the last layer of the network, allowing for task drift over time [44].

We assume inputs at time t (e.g., a history of data) are written $\boldsymbol{x}_t$ and outputs are written as $\boldsymbol{y}_t$. Our objective is to infer the probability of $\boldsymbol{y}_t$ conditional on $\boldsymbol{x}_t$. We write our predictive model as $\boldsymbol{y}_t = \Phi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)\boldsymbol{w}_t + \boldsymbol{\varepsilon}_t$, where $\Phi_{\boldsymbol{\theta}}(\cdot)$ is a matrix of neural network features parameterized by $\boldsymbol{\theta}$, $\boldsymbol{w}_t$ is a (time-varying) vector last layer, and $\boldsymbol{\varepsilon}_t$ is a zero mean Gaussian noise instantiation at time t with covariance Σ_ε (assumed independent across time). This differs from standard neural network regression in several ways. First, instead of a vector of features and a matrix last layer, we have a vector last layer and a matrix of features; this yields simpler inference within the model. Second, the last layer is assumed time-varying; we choose parameter dynamics

$$\boldsymbol{w}_{t+1} = A_{\boldsymbol{\theta}}(\boldsymbol{x}_t)\boldsymbol{w}_t + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) + \boldsymbol{\nu}_t \tag{1}$$

to enable tractability of inference. We typically assume simple $A, \boldsymbol{b}$ in practice, such as the identity matrix for A. The term $\boldsymbol{\nu}_t$ is a noise term, with variance Σ_ν. There are several alternate choices that can be made in this framework while retaining inferential tractability, discussed in depth in [43].

Inference within this model is as follows. Similar to Kalman filtering for state estimation, filtering consists of a prediction step—where the dynamics are applied to predict how the parameters change forward in time—and correction, where a measurement is used to update the estimate.

The prediction step is

$$\begin{aligned} \bar{\boldsymbol{w}}_{t+1|t} &= A_t \bar{\boldsymbol{w}}_{t|t} + \boldsymbol{b}_t \\ S_{t+1|t} &= A_t S_{t|t} A_t + \Sigma_\nu, \end{aligned} \tag{2}$$

where the subscript $t+1|t$ denotes the prediction for the value of the parameter at time $t+1$ made at time t. In the above, A_t is shorthand for $A(\boldsymbol{x}_t)$, which we will use for other quantities. The terms $\bar{\boldsymbol{w}}$ and S correspond to the mean and variance of the Gaussian prediction.

The correction step is

$$\begin{aligned} P_{t+1} &= \Phi_{t+1} S_{t+1|t} \Phi_{t+1}^\top + \Sigma_\varepsilon \\ K_{t+1} &= S_{t+1|t} \Phi_{t+1}^\top P_{t+1}^{-1} \\ \boldsymbol{e}_{t+1} &= \boldsymbol{y}_{t+1} - \Phi_{t+1} \bar{\boldsymbol{w}}_{t+1|t} \\ \bar{\boldsymbol{w}}_{t+1|t+1} &= \bar{\boldsymbol{w}}_{t+1|t} + K_{t+1} \boldsymbol{e}_{t+1} \\ S_{t+1|t+1} &= S_{t+1|t} - K_{t+1} \Phi_{t+1} S_{t+1|t}. \end{aligned} \tag{3}$$

From this framework, the predictive loss is the log likelihood of the prediction using mean and variance $\bar{\boldsymbol{w}}_{t+1|t}, S_{t+1|t}$. Critical to the meta-learning formulation, the parameters of the feature matrix, (possibly) the parameter dynamics matrices, and the noise covariances may be trained by back-propagating through the iterated inference and prediction, using the log-likelihood over a segment of an episode as a loss. This allows the model to learn features that are capable of adapting to novel environments, assuming the model is trained on a sufficiently diverse dataset.

IV. Problem Formulation

Our core problem formulation follows that of trajectory forecasting. Namely, we wish to generate plausible trajectory distributions for a time-varying number $N(t)$ of agents $A_1, \ldots, A_{N(t)}$. At time t, given the state $\boldsymbol{s}_i^{(t)} \in \mathbb{R}^D$ of each agent (e.g., relative position, velocity, acceleration, and heading), their histories for the previous H timesteps, and optional scene context $I^{(t)} \in \mathcal{I}$ (e.g., a local semantic map patch), we seek a distribution over each agents' future states for the next T timesteps $\boldsymbol{y}_t = \boldsymbol{s}_{1,\ldots,N(t)}^{(t+1:t+T)} \in \mathbb{R}^{T \times N(t) \times D}$, which we denote as $p(\boldsymbol{y}_t \mid \boldsymbol{x}_t)$, where $\boldsymbol{x}_t = (\boldsymbol{s}_{1,\ldots,N(t)}^{(t-H:t)}, I^{(t)}) \in \mathbb{R}^{(H+1) \times N(t) \times D} \times \mathcal{I}$ contains both the agent state histories and (optional) scene context $I^{(t)}$.

At training time, we assume access to a dataset $\mathcal{D}_\mathcal{S} = \{\boldsymbol{x}_j, \boldsymbol{y}_j\}_{j=1}^{|\mathcal{D}_\mathcal{S}|}$, collected from a set of source environments $\mathcal{S}$, and aim to obtain a model $p(\boldsymbol{y} \mid \boldsymbol{x})$ that can be effectively deployed to a target environment $\mathcal{T}$. Since this work tackles methods for adaptation, we focus on the following two problem settings that reflect the types of domain transfer commonly considered in practice: *Online*, where methods must adapt to *streaming* data (i.e., agents' current states $\boldsymbol{s}_{1,\ldots,N(t)}^{(t)}$ and optional scene context $I^{(t)}$ are observed at every timestep t), and *Offline*, where prediction methods have access to a small amount of *already-collected* data $\mathcal{D} \subset \mathcal{D}_\mathcal{T}$ from the target environment $\mathcal{T}$ with which they can finetune before deployment.

V. Generalizing Prediction Models Through Adaptive Meta-Learning

A. Architecture

Our architecture takes an encoder-decoder structure, similar to that of Trajectron++ [11]. Inputs (e.g., agent histories $\boldsymbol{x}$, encoded scene context I) are mapped to an overall scene encoding vector $\mathbf{v}$ via an attentional graph-structured recurrent network. The encoding $\mathbf{v}$ is then fed to the recurrent decoder, which maps $\mathbf{v}$ and the current hidden state h_t, as well as input states of the ego agent and other agents in the environment $\boldsymbol{s}_{1,\ldots,N(t)}^{(t)}$, to a distribution over actions taken by the ego agent. This action is then sampled and passed through the chosen dynamics model to generate the next state, which can then be fed back into the decoder at the next timestep.

Whereas Trajectron++ modeled multimodality via discrete latent variables [11], our model accounts for multimodality via controllable *aleatoric* uncertainty, sampling, and the recurrent network dynamics. We parameterize both output features Φ_t and the predictive noise covariance Σ_t as a function of the RNN hidden state. In particular, $\Phi_t = \Phi(h_t)$ and $\Sigma_t = \Sigma_\varepsilon(h_t)$, where h_t is the decoder's hidden state. This parameterization of the noise covariance as a function of history is important: it captures irreducible aleatoric uncertainty as a function of state (or state history).

In contrast to epistemic uncertainty, aleatoric uncertainty is assumed *irreducible*. For example, there is always some uncertainty as to the intentions of a car at an intersection, which can not be reduced through further observation (in contrast to epistemic uncertainty). In our model, this uncertainty will be unimodal for the first prediction step, but feeding output samples back through the recurrent decoder generates multimodality, yielding an overall multimodal output. Crucially, this controllable aleatoric uncertainty pairs naturally with our adaptive last layer: the higher the irreducible uncertainty in a state transition, the less information can be extracted from it for reducing epistemic uncertainty.

B. Multi-step Prediction and Loss

In this subsection, we detail our model's prediction procedure and loss computation.

Prediction. For multi-step prediction, we begin by sampling from the last layer, via $\boldsymbol{w}_{t+1|t}^i \sim \mathcal{N}(\bar{\boldsymbol{w}}_{t+1|t}, S_{t+1|t})$ for samples $i = 1, \ldots, N$. We then set $\tau = 0$ and repeat:

- Sample $\boldsymbol{u}_{t+\tau|t}^i \sim \mathcal{N}(\Phi_{t+\tau}^i \bar{\boldsymbol{w}}_{t+\tau|t}^i, \Sigma_{t+\tau}^i)$
- Compute next state $\boldsymbol{s}_{t+\tau+1|t}^i = f(\boldsymbol{s}_{t+\tau|t}^i, \boldsymbol{u}_{t+\tau|t}^i)$
- Set $h_{t+\tau+1}^i \leftarrow \text{DecoderCell}(\boldsymbol{s}_{t+\tau+1|t}^i, h_{t+\tau})$
- Sample $\bar{\boldsymbol{w}}_{t+\tau+1|t}^i \sim \mathcal{N}(\bar{\boldsymbol{w}}_{t+\tau|t}^i, \Sigma_\nu)$
- Set $\tau \leftarrow \tau + 1$

Looping this procedure for T steps will give N samples of T-length forecasts. Gradients may be computed through the sampling via the reparameterization trick [45].

Loss computation. We propose to approximate the predictive density via kernel density estimation [52]. In particular, we aim to maximize

$$\frac{1}{T} \sum_{\tau=t}^{T} \log \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(\boldsymbol{s}_{t+\tau+1}; \boldsymbol{s}_{t+\tau+1|t}^i, V_{t+\tau+1}^i), \tag{4}$$

where $\boldsymbol{s}_{t+\tau+1}$ corresponds to the true state at time t and $V_{t+\tau+1}^i$ corresponds to the covariance matrix used at time $t+\tau+1$ for particle i. Note that this loss can be implemented stably as the log-sum-exp of the log Gaussian density, and that the choice of $V_{t+\tau+1}^i$ is important. If $V_{t+\tau+1}^i$ is fixed across time, we risk later samples inducing extremely large variance in our gradient estimation. This

TABLE I

AVERAGE DISPLACEMENT ERROR (M) OBTAINED WHEN TRAINING AND EVALUATING METHODS ACROSS SCENES IN THE ETH/UCY PEDESTRIAN DATASETS. A, B, C, D, AND E DENOTE ETH, HOTEL, UNIV, ZARA1, AND ZARA2. RESULTS ON FDE CAN BE FOUND IN [46].

	A2B	A2C	A2D	A2E	B2A	B2C	B2D	B2E	C2A	C2B	C2D	C2E	D2A	D2B	D2C	D2E	E2A	E2B	E2C	E2D	AVG
S-STGCNN [47]	1.83	1.58	1.30	1.31	3.02	1.38	2.63	1.58	1.16	0.70	0.82	0.54	1.04	1.05	0.73	0.47	0.98	1.09	0.74	0.50	1.22
PECNet [48]	1.97	1.68	1.24	1.35	3.11	1.35	2.69	1.62	1.39	0.82	0.93	0.57	1.10	1.17	0.92	0.52	1.01	1.25	0.83	0.61	1.31
RSBG [49]	2.21	1.59	1.48	1.42	3.18	1.49	2.72	1.73	1.23	0.87	1.04	0.60	1.19	1.21	0.80	0.49	1.09	1.37	1.03	0.78	1.38
Tra2Tra [50]	1.72	1.58	1.27	1.37	3.32	1.36	2.67	1.58	1.16	0.70	0.85	0.60	1.09	1.07	0.81	0.52	1.03	1.10	0.75	0.52	1.25
SGCN [51]	1.68	1.54	1.26	1.28	3.22	1.38	2.62	1.58	1.14	0.70	0.82	0.52	1.05	0.97	0.80	0.48	0.97	1.08	0.75	0.51	1.22
T-GNN [24]	1.13	1.25	0.94	1.03	2.54	1.08	2.25	1.41	**0.97**	0.54	0.61	**0.23**	**0.88**	0.78	0.59	**0.32**	**0.87**	0.72	0.65	**0.34**	0.96
K_0	0.35	0.69	0.60	0.43	0.83	**0.58**	0.49	0.33	1.01	**0.39**	**0.40**	0.36	1.05	0.57	0.59	0.57	1.01	0.35	0.50	0.51	0.58
Ours	**0.33**	**0.56**	**0.50**	**0.38**	**0.80**	0.60	**0.43**	**0.31**	1.03	0.41	0.41	0.38	0.93	**0.32**	**0.48**	0.35	0.91	**0.31**	**0.49**	0.44	**0.52**

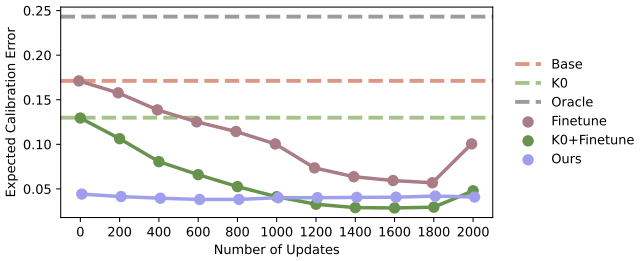

Fig. 2. [Zara1 → Hotel] Our method learns a well-calibrated prior, and maintains its calibration as it observes more data. Gradient-based finetuning improves calibration at first, but yields overconfidence in later steps. Results on additional metrics can be found in the appendix.

is a natural consequence of the outcome space for the integrator dynamics being wider for long horizons than short horizons, and the uncertainty metric in KDE should reflect this. Accordingly, we propose to integrate the uncertainty via $V^i_{t+\tau+1} = \sum_{k=t}^{t+\tau+1} \Sigma^i_k$. In the case where the dynamics are a single integrator (a common choice in Trajectron++ [11]), this captures the state uncertainty induced by aleatoric control uncertainty exactly, and thus it naturally scales in time.

C. Adaptation

While we have described a multi-step prediction formulation, we will only use one step predictions for adaptation. Because the first prediction is unimodal and Gaussian, adaptation may be performed using exact inference on the last layer. While this choice reduces the maximum theoretical performance of our approach, instead of *approximately* adapting our model to the multi-step prediction error, we perform *exact* inference based on the single step predictions. Most importantly, this trade-off enables high execution frequencies due to the simplicity of the update and high data efficiency of exact inference. Further discussion of computational complexity is available in the appendix.

Online Adaptation. In the online adaptation setting, an agent makes a probabilistic prediction, for which the initial prediction for all particles is the same. The mean and variance of this prediction are used to update the last layer, which corresponds exactly to the update equations described in Section III. Critically, adaptation in this setting is done *per agent* (as each agent has only local information) and with temporally correlated data.

Offline Adaptation. In offline adaptation, a small dataset from the target environment is assumed to be provided. This dataset, which consists of several different agents' state information, is then used to adapt the last layer via the same adaptation mechanism.

Combining Last Layer Adaptation with Gradient-Based Finetuning. We can also combine our approach with gradient-based finetuning for even more efficient and effective domain transfer in the offline setting. In particular, we first perform M steps of last layer adaptation and then switch to gradient-based finetuning for future updates. This sequential update scheme leverages the fast initial adaptation of our last layer exact inference, while exploiting the high capacity and strong performance of gradient-based fine-tuning [53]. This combination of exact inference for efficiency combined with gradient-based fine-tuning for capacity has been exploited previously [54], although not purely sequentially.

VI. EXPERIMENTS

We evaluate our method[1] on a variety of transfer scenarios from real-world pedestrian and vehicle data, comparing against prior works and ablations of our method on multiple deterministic and probabilistic metrics. Our approach was implemented in PyTorch [55] and all datasets were interfaced through `trajdata` [56], a recently-released unified interface to trajectory forecasting datasets.

Datasets. We use the ETH [57] and UCY [58] pedestrian datasets, as well as the nuScenes [59] and Lyft Level 5 [60] autonomous driving datasets. The ETH and UCY datasets are a standard benchmark in the field, comprised of pedestrian trajectories captured at 2.5 Hz ($\Delta t = 0.4s$) in Zurich and Cyprus, respectively. In total, there are 5 sets of data from 4 unique environments, containing a total of 1536 pedestrians.

nuScenes is a large-scale autonomous driving dataset comprised of 1000 scenes collected in Boston and Singapore. Each scene is annotated at 2 Hz ($\Delta t = 0.4s$) and is $20s$ long. Lyft Level 5 is comprised of 170K scenes collected in Palo Alto, each of which are annotated at 10 Hz ($\Delta t = 0.1s$) and roughly $25s$ long. While this equates to more than 1000 hours of driving data, it was only captured along a single route 6.8 miles long, yielding significant overlap across scenes in the dataset. As a result, we instead evaluate on the Lyft Level 5 sample split, a 100-scene subset of the dataset.

Ablations and Oracle. We compare against the following five ablations as well as an oracle: (1) *Base* is the original Trajectron++ [11] model without any of our adaptive architecture; (2) *Finetune* is Base combined with gradient-based finetuning for adaptation, using an initial learning rate $10\times$ smaller than Base, as is common in practice. (3) K_0 is our method without any adaptive training, i.e., always setting $\bar{\boldsymbol{w}}_{t|t} = \bar{\boldsymbol{w}}_{0|0}$ and $S_{t|t} = S_{0|0}$ in Eq. (2) and not applying corrections; (4) K_0+*Finetune* is the K_0 baseline combined with gradient-based finetuning. Note, only the last layer is finetuned (the rest of the model is frozen).

[1]Code and models will be made available at `https://github.com/nvr-avg/adaptive-prediction`.

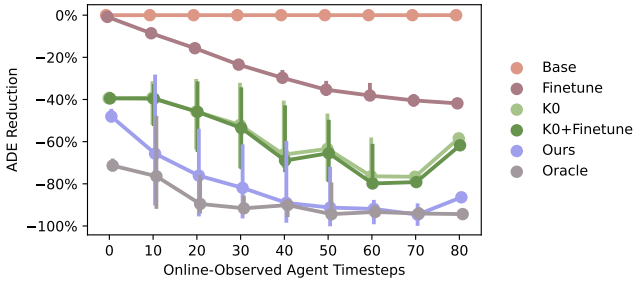

Fig. 3. [Zara1 → Hotel] Our method's prediction accuracy improves rapidly as it observes data online, significantly outperforming naïve transfer and other ablations, even matching the oracle. Error bars are 95% CIs, lower is better. Results on additional metrics can be found in the appendix.

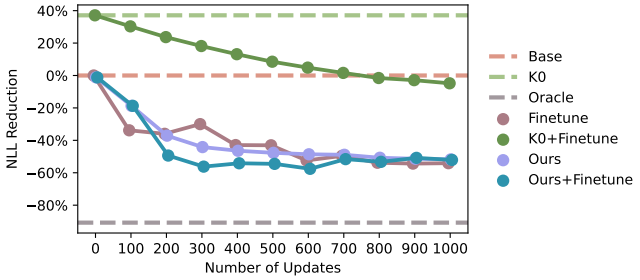

Fig. 4. [nuScenes → Lyft] Even in the face of significant domain shift (i.e., 5× timestep frequency and different map annotations), our method's predictions improve smoothly with more data, outperforming whole-model fine-tuning after only 200 data samples. Results on additional metrics can be found in the appendix.

In essence, this ablation compares using gradient descent for adaptation instead of the correction step in Eq. (3); and (5) *Ours+Finetune* is our full method combined with gradient-based finetuning, as described in Section V-C. In our experiments, we switch to gradient-based finetuning after $M = 100$ Bayesian last layer updates. Finally, since our experiments focus on transferring models from one environment to another, we also show the performance of an oracle, i.e., the base Trajectron++ [11] model trained on data from the target environment, representing ideal transfer performance. Additional baselines will be described in their respective sections.

Metrics. We evaluate prediction accuracy with a variety of deterministic and probabilistic metrics: *Average/Final Displacement Error (ADE/FDE)*: mean/final ℓ_2 distance between the ground truth (GT) and predicted trajectories; *minADE*$_{5/10}$: ADE/FDE between the GT and best of 5 or 10 samples, respectively; and *Negative Log-Likelihood (NLL)*: mean NLL of the GT trajectory under the predicted distribution. In addition to prediction accuracy, we also evaluate methods' predictive calibration by computing their Expected Calibration Error (ECE) [61], which compares how closely a model's predictive uncertainty matches its empirical uncertainty, as defined by the fraction of GT future positions lying within specified probability thresholds.

A. Pedestrian Data

Evaluation Protocol. As in recent prior work [24], we treat each of the 5 scenes in the ETH and UCY datasets as individual source domains $\mathcal{S} \in \{\texttt{ETH}, \texttt{Hotel}, \texttt{Univ}, \texttt{Zara1}, \text{and } \texttt{Zara2}\}$, and use the other 4 scenes as our target domains $\mathcal{T} \neq \mathcal{S}$, yielding 20 cross-domain pairings. To ensure that we are purely evaluating adaptation, we

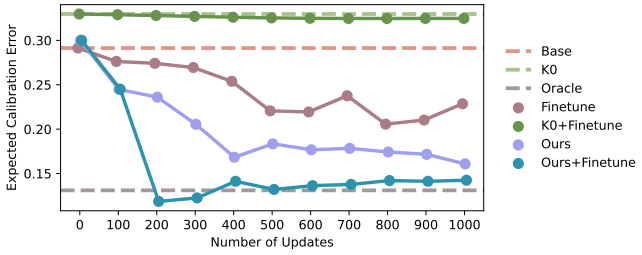

Fig. 5. [nuScenes → Lyft] While our method's calibration already improves faster than baselines, combining it with gradient-based finetuning significantly accelerates improvement, almost immediately yielding a better calibrated model than the oracle.

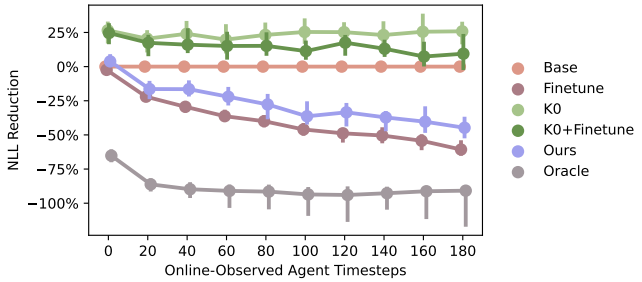

Fig. 6. [nuScenes → Lyft] Our method's online NLL reduction tracks closely to that of whole-model finetuning, while being much less computationally expensive. Results on additional metrics can be found in the appendix.

restrict models' training sets to *only* be their source domain's training split $\mathcal{D}_{\mathcal{S},\text{train}}$ (leaving the source domain's validation set $\mathcal{D}_{\mathcal{S},\text{val}}$ for hyperparameter tuning) and evaluate their performance on the entire target dataset $\mathcal{D}_{\mathcal{T}}$. This is a notable difference from the evaluation protocol proposed in [24], where prediction methods train on the source domain's training split $\mathcal{D}_{\mathcal{S},\text{train}}$ *as well as the target domain's validation split* $\mathcal{D}_{\mathcal{T},\text{val}}$. This complicates measuring a model's capability for domain adaptation, as it has already seen data from the target environment during training. In the rest of this section, up to $3.2s$ of history are observed and the next $4.8s$ are predicted, as in prior works [24].

Scene-to-Scene Transfer. In this setting, we train on a source scene X and directly transfer to a target scene Y (denoting the pair as "X2Y"), without any online or offline adaptation to fairly compare with prior work. We additionally compare to a large variety of state-of-the-art approaches, namely S-STGCNN [47], PECNet [48], RSBG [49], Tra2Tra [50], SGCN [51], and T-GNN [24]. As can be seen in Table I, our K_0 ablation significantly outperforms all baseline approaches on the vast majority of transfer settings, showing that our prior and its analytical propagation of uncertainty already yields strong performance. Our full method further improves upon K_0 due to its additional conditioning on observed history (i.e., using Eq. (3) with the agent's H-length trajectory history in each data sample).

In the ETH/UCY datasets, scenes primarily differ due to their geographic regions (e.g., Zurich and Cyprus) and layout (i.e., a university campus and an urban sidewalk). Accordingly, transfer asymmetries can be seen in Table I, e.g., training on the ETH - Univ scene and transferring to an urban sidewalk in Cyprus (A2D and A2E) yields much lower errors than the reverse direction (D2A and E2A).

Offline Calibration. Fig. 2 demonstrates that our method learns a well-calibrated prior (Ours at 0 updates), and main-

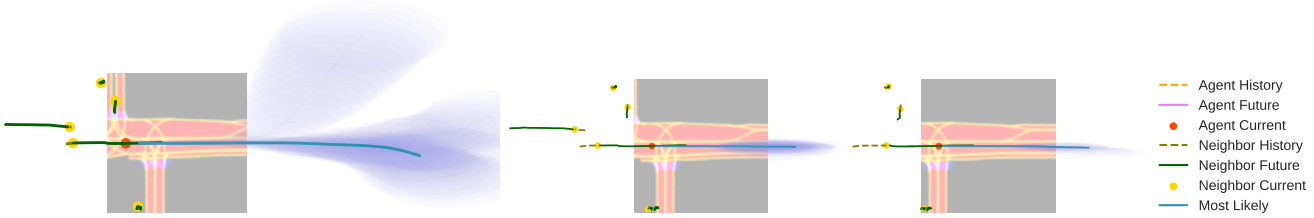

Fig. 7. [nuScenes → Lyft] **Left**: Initially, before making any adaptive steps, our model's prior predictions are uncertain and spread out (primarily due to the significant domain shift between nuScenes and Lyft). After observing only $0.5s$ of data online (**Middle**), it has adapted to the environment and its predictions better match the GT future. Finally, after observing $1s$ of data (**Right**), our method has tightly clustered its spatial probability around the GT.

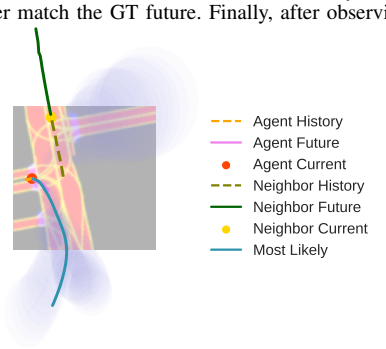

Fig. 8. [nuScenes → Lyft] Our multi-step sampling strategy yields multimodal predictions. 5 samples are shown here, as well as the most-likely prediction.

tains its calibration as it observes more data offline. Further, while gradient-based finetuning improves calibration at first, it yields overconfidence in later steps (seen as an increase in ECE after 2000 updates). The oracle is poorly calibrated because Trajectron++ [11] tends to be overconfident when predicting pedestrians (detailed calibration results showing this can be found in the appendix). Accurate calibration is critical to designing closed-loop autonomy stacks that are not over- or under-conservative in their decision making, and is thus vital for safe autonomy.

Online Adaptation. Focusing on the transfer from UCY-Zara1 to ETH-Hotel (D2B), Fig. 3 shows that our method's median ADE reduction rapidly approaches the oracle as more of an agent's trajectory is observed online. Both Finetune and K_0+Finetune are slow to adapt online, with Finetune only reducing ADE by 40% in the same time our method achieves 90%+ reductions and K_0+Finetune barely differing from the non-adaptive K_0. Additional results on the other metrics can be found in the Appendix.

B. Autonomous Driving Data

Evaluation Protocol. We treat nuScenes as the source domain and Lyft Level 5 as the target domain. In particular, we train models on the nuScenes prediction challenge `train` split, tune hyperparameters on the `train_val` split, and evaluate methods' capability to adapt to the entire Lyft Level 5 sample split. This is a *much* more challenging domain transfer problem than in the ETH/UCY datasets, since the nuScenes and Lyft datasets feature different underlying map annotations (e.g., nuScenes does not annotate lanes through intersections whereas Lyft does) and data frequencies (2 Hz vs 10 Hz), in addition to being collected in diverse cities with unique road geometries. In the rest of this section, up to $2s$ of history are observed and the next $6s$ are predicted, as in the nuScenes prediction challenge.

Offline Transfer. As can be seen in Fig. 4, our method rapidly reduces NLL, outperforming Finetune after receiving only 200 updates. Ours+Finetune further improves upon our method, showing that gradient-based finetuning is a complementary and performant addition to our last layer adaptation scheme for offline transfer.

Offline Calibration. Fig. 5 confirms that our approach's predictive calibration improves with more data. Switching to gradient-based finetuning after 100 last-layer adaptation steps significantly accelerates improvement, even yielding a better calibrated model than the oracle.

Online Adaptation. Fig. 6 reinforces that our method's performance rapidly improves as it sequentially observes data online. While whole-model finetuning (Finetune) is slightly more performant, it is not real-time feasible, further highlighting our method's performance as it tracks closely to Finetune while being significantly less computationally expensive.

Qualitative Results. Fig. 7 shows our model's adaptation visually. Before seeing any data, our model's predictions are spread out, uncertain, and overshoot the GT (due to the nuScenes-Lyft data frequency mismatch). After observing only $0.5s$ of data, our model has adapted to the data frequency mismatch and its predictions better match the GT future. Finally, after observing $1s$ of data, our method has tightly clustered its spatial probability around the GT, yielding an accurate prediction with confident associated uncertainty. Fig. 8 shows that our proposed sampling scheme yields diverse, multimodal predictions in an unusual intersection in Palo Alto.

VII. Discussion and Conclusion

In this work, we have presented a model that combines the strength of recurrent models with adaptive, optimization-based meta-learning. This combination both adapts efficiently and yields strong performance. Our approach has shown particularly strong results for the one-to-one transfer setting. One strength of the approach presented in this paper—which we do not address due to space constraints—is the ability to do effective many-to-one transfer, in which many training datasets are available. This setting is reflective of deploying in a new geographical location, where the full training dataset of many other cities is available for pre-training. We anticipate that further training data diversity will result in monotonic improvement to performance, yielding both better calibrated priors and more expressive learned features. Indeed, careful evaluation of the scaling performance with respect to meta-training data is an important direction of future work.

REFERENCES

[1] General Motors, "Self-driving safety report," 2018, Available at https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf.

[2] Uber Advanced Technologies Group, "A principled approach to safety," 2020, Available at https://uber.app.box.com/v/UberATGSafetyReport.

[3] Lyft, "Self-driving safety report," 2020, Available at https://2eg1kz1onwfq1djllo2xh4bb-wpengine.netdna-ssl.com/wp-content/uploads/2020/06/Safety_Report_2020.pdf.

[4] Argo AI, "Developing a self-driving system you can trust," Apr. 2021, Available at https://www.argo.ai/wp-content/uploads/2021/04/ArgoSafetyReport.pdf.

[5] Motional, "Voluntary safety self-assessment," 2021, Available at https://drive.google.com/file/d/1JjfQByU_hWvSfkWzQ8PK2ZOZfVCqQGDB/view.

[6] Zoox, "Safety report volume 2.0," 2021, Available at https://zoox.com/safety/.

[7] NVIDIA, "Self-driving safety report," 2021, Available at https://images.nvidia.com/content/self-driving-cars/safety-report/auto-print-self-driving-safety-report-2021-update.pdf.

[8] R. Bellan, "Cruise starts mapping Dubai's streets in prep for 2023 robotaxi launch," 2022, Available at https://techcrunch.com/2022/07/24/cruise-starts-mapping-dubais-streets-in-prep-for-2023-robotaxi-launch/.

[9] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022.

[10] J. Harrison, A. Sharma, and M. Pavone, "Meta-learning priors for efficient online bayesian regression," in *Workshop on Algorithmic Foundations of Robotics*, 2018.

[11] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conf. on Computer Vision*, 2020.

[12] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *Int. Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[13] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Conf. on Neural Information Processing Systems*, 2015.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Conf. on Neural Information Processing Systems*, 2014.

[15] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. H. Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks," in *Conf. on Neural Information Processing Systems*, 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Conf. on Neural Information Processing Systems*, 2017.

[17] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *IEEE Int. Conf. on Pattern Recognition*, 2020.

[18] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *IEEE Int. Conf. on Computer Vision*, 2021, pp. 9813–9823.

[19] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[20] A. Kamenev, L. Wang, O. B. Bohan, I. Kulkarni, B. Kartal, A. Molchanov, S. Birchfield, D. Nistér, and N. Smolyanskiy, "PredictionNet: Real-time joint probabilistic traffic prediction for planning, control, and simulation," in *Proc. IEEE Conf. on Robotics and Automation*, 2022.

[21] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Uncertainty estimation for cross-dataset performance in trajectory prediction," 2022, Available at https://arxiv.org/abs/2205.07310.

[22] F. Marchetti, F. Becattini, L. Seidenari, and A. Del Bimbo, "Multiple trajectory prediction of moving agents with memory augmented networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2020.

[23] C. Xu, W. Mao, W. Zhang, and S. Chen, "Remember intentions: Retrospective-memory-based trajectory prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2022.

[24] Y. Xu, L. Wang, Y. Wang, and Y. Fu, "Adaptive trajectory prediction via transferable GNN," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2022.

[25] L. Wang, Y. Hu, L. Sun, W. Zhan, M. Tomizuka, and C. Liu, "Transferable and adaptable driving behavior prediction," 2022, Available at https://arxiv.org/abs/2202.05140.

[26] Q. Song, X. Zhao, Z. Feng, and B. Song, "Recursive least squares algorithm with adaptive forgetting factor based on echo state network," in *IEEE World Congress on Intelligent Control and Automation*, 2011.

[27] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*. Dover Publications, 2014.

[28] Y. Cheng, W. Zhao, C. Liu, and M. Tomizuka, "Human motion prediction using semi-adaptable neural networks," in *American Control Conference*, 2019.

[29] W. Si, T. Wei, and C. Liu, "AGen: Adaptable generative prediction networks for autonomous driving," in *IEEE Intelligent Vehicles Symposium*, 2019.

[30] A. Abuduweili and C. Liu, "Robust online model adaptation by extended kalman filter with exponential moving average and dynamic multi-epoch strategy," in *Learning for Dynamics & Control*, 2020.

[31] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys*, vol. 53, no. 3, pp. 1–34, 2020.

[32] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Int. Conf. on Machine Learning*, 2017.

[33] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Conf. on Neural Information Processing Systems*, vol. 30, 2017.

[34] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2021, early access.

[35] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv:1611.02779*, 2016.

[36] H. Edwards and A. Storkey, "Towards a neural statistician," *arXiv:1606.02185*, 2016.

[37] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," *Int. Conf. on Learning Representations*, 2017.

[38] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Int. Conf. on Machine Learning*, 2016, pp. 1842–1850.

[39] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv:1611.05763*, 2016.

[40] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *International conference on artificial neural networks*, 2001, pp. 87–94.

[41] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv:1803.02999*, 2018.

[42] J. Harrison, A. Sharma, C. Finn, and M. Pavone, "Continuous meta-learning without tasks," in *Conf. on Neural Information Processing Systems*, 2020, submitted.

[43] J. Harrison, "Uncertainty and efficiency in adaptive robot learning and control," Ph.D. dissertation, Stanford University, Dept. of Mechanical Engineering, 2021.

[44] M. West and J. Harrison, *Bayesian forecasting and dynamic models*. Springer Science & Business Media, 2006.

[45] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, Available at https://arxiv.org/abs/1312.6114.

[46] B. Ivanovic, J. Harrison, and M. Pavone. (2022) Expanding the deployment envelope of behavior prediction via adaptive meta-learning. Available at http://www.borisivanovic.com/files/icra_extended.pdf.

[47] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-STGCNN: A social spatiotemporal graph convolutional neural network for human trajectory prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[48] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destinaton: Endpoint conditioned trajectory prediction," in *European Conf. on Computer Vision*, 2020.

[49] J. Sun, Q. Jiang, and C. Lu, "Recursive social behavior graph for trajectory prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[50] Y. Xu, D. Ren, M. Li, Y. Chen, M. Fan, and H. Xia, "Tra2Tra: Trajectory-to-trajectory prediction with a global social spatial-temporal attentive neural network," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1574–1581, 2021.

[51] L. Shi, L. Wang, C. Long, S. Zhou, M. Zhou, Z. Niu, and G. Hua, "SGCN: Sparse graph convolution network for pedestrian trajectory

prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2021.
[52] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.
[53] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv:1801.06146*, 2018.
[54] J. Willes, J. Harrison, A. Harakeh, C. Finn, M. Pavone, and S. Waslander, "Bayesian embeddings for few-shot open world recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2022.
[55] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017.
[56] B. Ivanovic and contributors. (2022) trajdata: A unified interface to many trajectory forecasting datasets. Available at https://github.com/nvr-avg/trajdata.
[57] S. Pellegrini, A. Ess, K. Schindler, and L. v. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *IEEE Int. Conf. on Computer Vision*, 2009.
[58] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.
[59] 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 *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.
[60] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," in *Conf. on Robot Learning*, 2020.
[61] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *Conf. on Neural Information Processing Systems*, vol. 32, 2019.
[62] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone, "Safe active dynamics learning and control: A sequential exploration-exploitation framework," *IEEE Transactions on Robotics*, 2022, in Press.

APPENDIX

A. Additional Pedestrian Evaluations

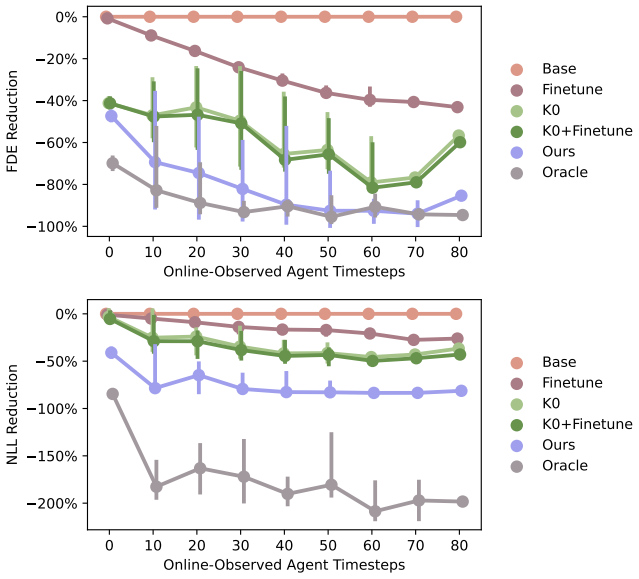

Fig. 9. peds per-agent online Caption. FDE and NLL median improvement.

B. Additional Autonomous Driving Evaluations

C. Additional Discussion

Computational Complexity. There are several considerations that are important for achieving efficient performance in the model. First, in practice, we fix a set of parameters for each output dimension, which are adapted independently. This corresponds to choosing a diagonal noise and parameter covariance, fixing parameter dynamics (A) diagonal (in practice we simply choose this to be identity), and fixing independent priors for parameters of each output dimension. Parameterizing each output dimension independently substantially reduces computational complexity, as discussed in [62].

Furthermore, we may implement the model such that complexity is at most quadratic in the parameter dimension. First, A must be chosen appropriately; identity is sufficient but other representations are possible. In addition, the multiplication of $K_{t+1}\Phi_{t+1}S_{t+1|t}$ should be done with the last two terms first, before multiplying by the gain matrix K_{t+1}. This enables better representational capacity for adaptation by enabling choice of a larger number of features.

Finally, we note that prediction in the model is sequential in time due to the unrolling of the recurrent model. Thus, at run time, parallelizing over a large number of predictions is straightforward. Indeed, for hardware such as GPUs or specialized cores for neural networks, parallelized matrix multiplications are extremely efficient, and thus many samples can be parallelized across.

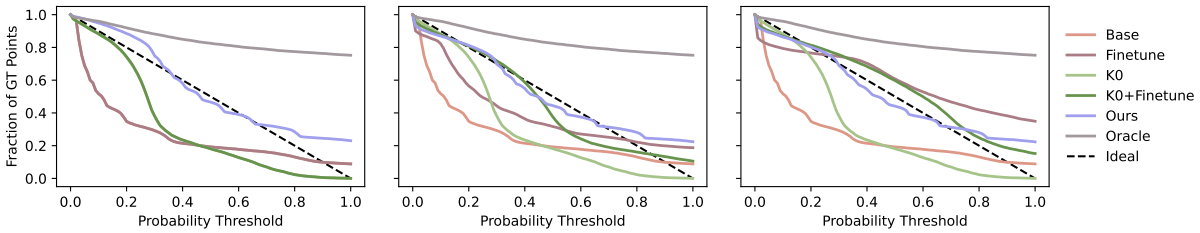

Fig. 10. peds calibration offline 1, 1000, 2000.

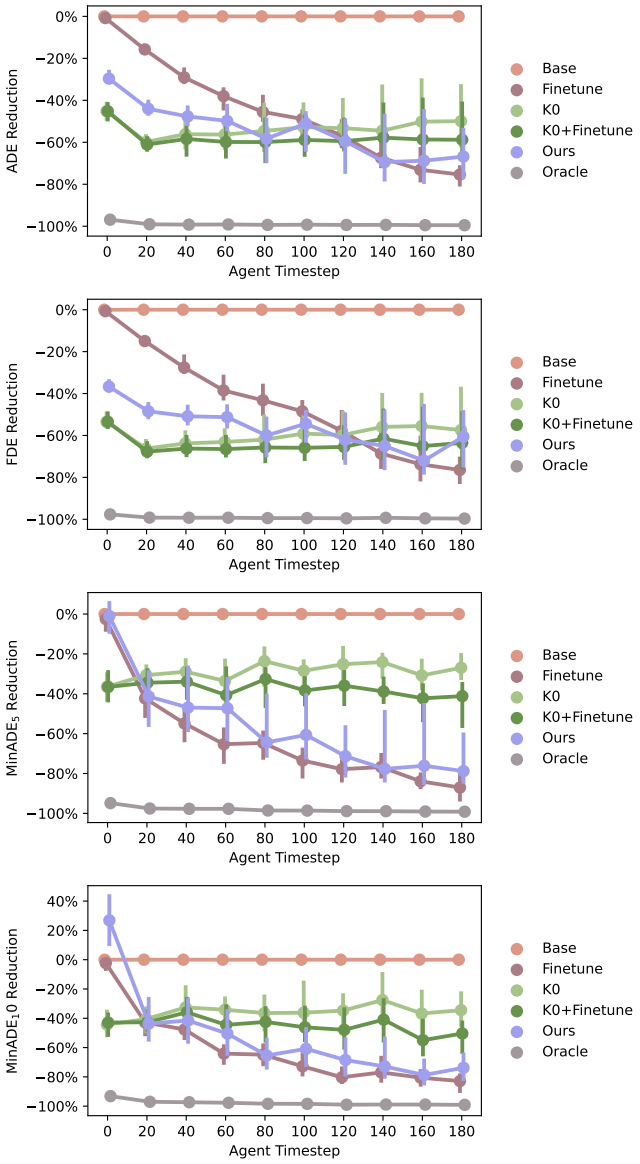

Fig. 11. lyft per-agent online. median improvement

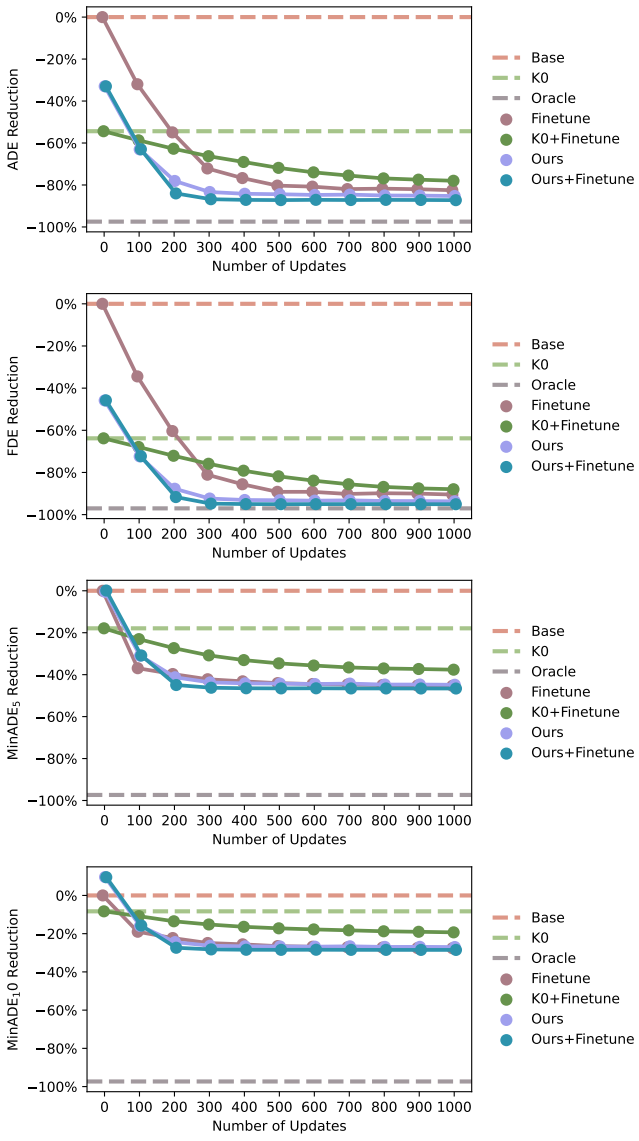

Fig. 12. lyft offline.

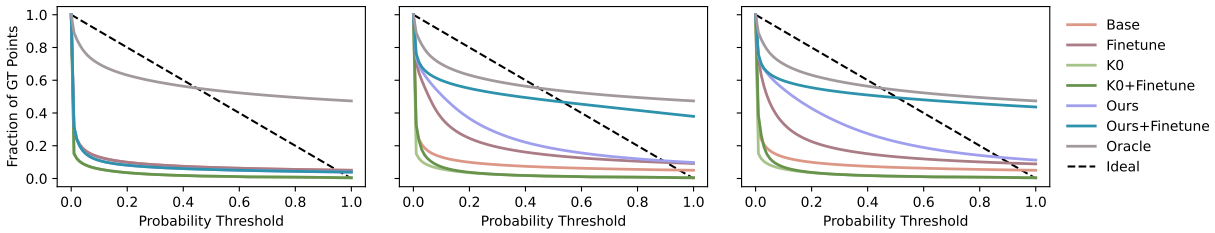

Fig. 13. lyft calibration offline 1, 500, 1000.