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

Boris Ivanovic[1] **James Harrison**[2] **Marco Pavone**[1,3]
[1]NVIDIA Research [2]Google Research, Brain Team [3]Stanford University
{bivanovic, mpavone}@nvidia.com, jamesharrison@google.com, pavone@stanford.edu

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. All of our code, models, and data are available at `https://github.com/NVlabs/adaptive-prediction`.

1 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 [1–7]. 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 standards, laws, agent types, and road geometries.

Contributions. Our key contributions are threefold. First, we combine ALPaCA-based [8] last layer adaptation with recurrent models, and show that they work synergistically to enable both broad generalization and efficient transfer. Second, we present a novel trajectory forecasting algorithm based on Trajectron++ [9] 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. 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.

2 Related Work

Adaptive Behavior Prediction. There have been a plethora of trajectory forecasting datasets released in recent years, with thousands of hours of data now publicly available [10]. Accordingly, there have been many methods tackling the problem of domain adaptation for trajectory forecasting. They can broadly be categorized into the following three groups: (1) Memory-based approaches generally tackle

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

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 [11, 12]. One key drawback, however, is that constantly-growing extra memory may not be feasible for platforms with fixed memory and compute limits. In contrast, our method does not require any extra memory. (2) Architectural methods generally employ neural networks whose intermediate representations or structures are generalizable and can transfer to different domains [13, 14]. A key drawback of such approaches is that they either require labeled data from the target domain during training [13] or are not inherently online-adaptive [14]. (3) Finally, least-squares-based methods typically employ recursive least squares to adapt models to data observed online. Similar to our approach, RLS-PAA [15, 16] 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 domain transfer).

Few-Shot Learning by Meta-Learning. 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 [17–24]. 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 [20, 24, 25]. 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 [17, 8, 26, 22], which directly leverage optimization in the inner learning loop, have been a major research focus. Our work 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 by showing that recurrence-based adaptation can be effectively paired with optimization-based meta-learning.

3 Generalizing Prediction Models Through Adaptive Meta-Learning

Problem Formulation. Our core problem formulation follows that of trajectory forecasting. Namely, we wish to generate plausible distributions $p(\boldsymbol{y}_t \mid \boldsymbol{x}_t, I^{(t)})$ of a time-varying number $N(t)$ of agents' future trajectories $\boldsymbol{y}_t = \boldsymbol{s}_{1,\dots,N(t)}^{(t+1:t+T)}$ given their past state histories $\boldsymbol{x}_t = \boldsymbol{s}_{1,\dots,N(t)}^{(t-H:t)}$ and (optional) scene context $I^{(t)}$. At train time, we assume access to a dataset $\mathcal{D}_{\mathcal{S}} = \{\boldsymbol{x}_j, \boldsymbol{y}_j\}$ 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: *Online*, where methods must adapt to *streaming* data (i.e., new inputs 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 $\mathcal{T}$ on which they can finetune.

Architecture. At a high-level, our method leverages an *adaptive* formulation of meta-learning, an overview of which can be found in Appendix A.1. Our architecture takes an encoder-decoder structure, similar to Trajectron++ [9]. Inputs (e.g., agent histories and any encoded scene context) are mapped to an overall scene encoding vector $\mathbf{v}$ via an attentional graph-structured recurrent network. $\mathbf{v}$ is then fed to the recurrent decoder, which maps $\mathbf{v}$, the current hidden state h_t, and input states of the observed agents in the environment $\boldsymbol{s}_{1,\dots,N(t)}^{(t)}$ to a distribution over actions taken by the predicted 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++ models multimodality via discrete latent variables [9], our model accounts for multimodality via controllable *aleatoric* uncertainty, sampling, and recurrent network dynamics. We parameterize both output features $\Phi_t = \Phi(h_t)$ and the predictive noise covariance $\Sigma_t = \Sigma_\varepsilon(h_t)$ as a function of the decoder's hidden state h_t. This parameterization as a function of history is important: it captures irreducible aleatoric uncertainty as a function of state (or state history).

Multi-step 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 $i = 1, \dots, N$. We then set $\tau = 0$ and repeatedly 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 the next state $\boldsymbol{s}_{t+\tau+1|t}^i = f(\boldsymbol{s}_{t+\tau|t}^i, \boldsymbol{u}_{t+\tau|t}^i)$, set $h_{t+\tau+1}^i \leftarrow \text{DecoderRNNCell}(\boldsymbol{s}_{t+\tau+1|t}^i, h_{t+\tau})$, sample the next $\bar{\boldsymbol{w}}_{t+\tau+1|t}^i \sim \mathcal{N}(\bar{\boldsymbol{w}}_{t+\tau|t}^i, \Sigma_\nu)$, and set $\tau \leftarrow \tau + 1$ for the next iteration. Looping this procedure for T steps gives N samples of T-length forecasts. Gradients are computed through the sampling via the reparameterization trick [27].

Table 1: Average Displacement Error (m) obtained when training and evaluating methods across scenes in the ETH/UCY pedestrian datasets. A, B, C, D, E denote ETH, Hotel, Univ, Zara1, Zara2.

	A2B	A2C	A2D	A2E	B2A	B2C	B2D	B2E	C2A	C2B	C2D	C2E	D2A	D2B	D2C	D2E	E2A	E2B	E2C	E2D	AVG
S-STGCNN [28]	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 [29]	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 [30]	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 [31]	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 [32]	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 [13]	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**

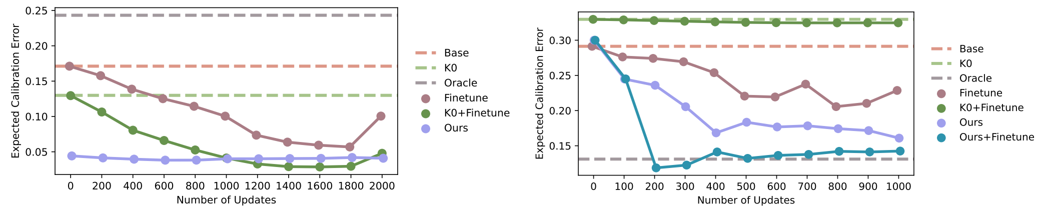


Figure 1: *Offline.* **Left:** [Zara1 $\rightarrow$ Hotel] Our method learns a well-calibrated prior, and maintains its calibration as it observes more data. Gradient-based finetuning improves calibration at first, but yields overconfidence in later steps. **Right:** [nuScenes $\rightarrow$ Lyft] While our method's calibration already improves faster than baselines, combining it with gradient-based finetuning significantly accelerates improvement, immediately yielding a better calibrated model than the oracle. Lower is better.

Loss. We propose to approximate the predictive density via kernel density estimation [33], maximizing $\frac{1}{T}\sum_{\tau=t}^{T}\log\frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(\boldsymbol{s}_{t+\tau+1};\boldsymbol{s}^i_{t+\tau+1|t},V^i_{t+\tau+1})$, where $\boldsymbol{s}_{t+\tau+1}$ is the true state at time t and $V^i_{t+\tau+1}$ corresponds to the covariance matrix used at time $t+\tau+1$ for particle i. Note that the choice of $V^i_{t+\tau+1}$ is important; If $V^i_{t+\tau+1}$ is fixed across time, later samples may induce extremely large variance in our gradient estimation. 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++ [9]), this captures the state uncertainty induced by aleatoric control uncertainty *exactly*.

Adaptation. In the online setting, we make a probabilistic prediction, the mean and variance of which are used to update the last layer using the update equations described in Appendix A.1. Critically, adaptation in this setting is done *per agent* (as each agent has only local information) and with temporally-correlated data. In the offline setting, 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 [34].

4 Experiments

Datasets. We evaluate our method on the ETH [35] and UCY [36] pedestrian datasets, as well as the nuScenes [37] and Lyft Level 5 [38] autonomous driving datasets. The ETH and UCY datasets are a standard benchmark in the field, comprised of pedestrian trajectories captured at 2.5 Hz in Zurich and Cyprus, respectively. As in prior work [13], up to $3.2s$ of history are observed and the next $4.8s$ are predicted. nuScenes is a large-scale autonomous driving dataset comprised of 1000 scenes annotated at 2 Hz collected in Boston and Singapore. Lyft Level 5 is comprised of 170K scenes annotated at 10 Hz collected in Palo Alto. As in the nuScenes prediction challenge, up to $2s$ of history are observed and the next $6s$ are predicted. Details about our experimental protocol can be found in Appendix A.2.

Ablations, Oracle, and Baselines. We compare against the following five ablations: (1) *Base* is the original Trajectron++ [9] model without any of our adaptive architecture; (2) *Finetune* is Base combined with gradient-based finetuning for adaptation. (3) K_0 is our method without any adaptation, i.e., only using the prior model; (4) K_0+*Finetune* is the K_0 baseline combined with

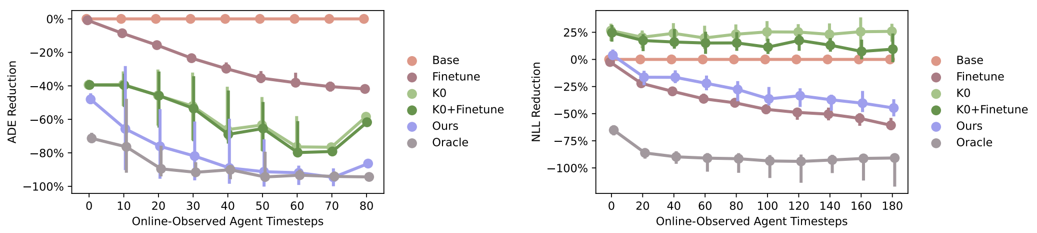


Figure 2: *Online.* **Left:** [Zara1 → Hotel] Our method's prediction accuracy improves rapidly as it observes data online, significantly outperforming naïve transfer and other ablations, even matching the oracle. **Right:** [nuScenes → Lyft] Our method's online NLL reduction tracks closely to that of whole-model finetuning, while being much less computationally expensive. All error bars are 95% CIs, lower is better. Results on additional metrics can be found in Appendices A.3 and A.4.

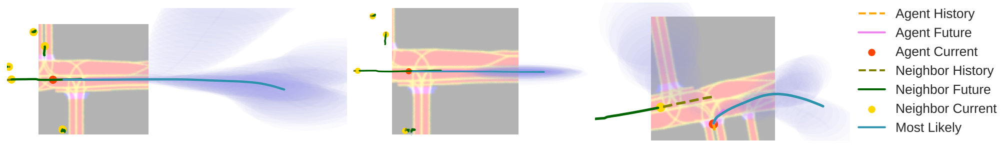


Figure 3: [nuScenes → Lyft] **Left**: Initially, before making any adaptive steps, our model's prior predictions are uncertain and spread out (due to the significant domain shift between nuScenes and Lyft, i.e., $5\times$ timestep frequency and different map annotations). **Middle**: After observing only $0.5s$ of data, it has adapted to the environment and its predictions better match the GT future. **Right**: Our multi-step sampling strategy yields multimodal predictions. Five samples are shown in all plots.

gradient-based finetuning. Note, only the last layer is finetuned (the rest of the model is frozen). In essence, this ablation compares using gradient descent for adaptation instead of Bayesian updates; and (5) *Ours+Finetune* is our full method combined with gradient-based finetuning, switching after $M = 100$ Bayesian last layer updates. Finally, we also include an oracle, i.e., Trajectron++ [9] trained on data from the target environment, representing ideal transfer performance.

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 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. We also evaluate methods' predictive calibration by computing their Expected Calibration Error (ECE) [39], comparing a model's predictive uncertainty with its empirical uncertainty.

Direct Transfer. In Table 1, 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 an assortment of state-of-the-art methods [28–32, 13]. As can be seen, our non-adaptive ablation (K_0) significantly outperforms other methods on the vast majority of transfer settings, showing that our prior and its analytical propagation of uncertainty already yield strong performance.

Offline Calibration. Fig. 1 shows that our method learns a well-calibrated prior for pedestrians, and maintains its calibration with more updates. Results on Lyft further highlight our method's improvement, and show that 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. Focusing on the transfer from UCY-Zara1 to ETH-Hotel (D2B), Fig. 2 (left) shows that our method's median ADE reduction rapidly approaches the oracle as more of an agent's trajectory is observed online. This performance is replicated in transferring from nuScenes to Lyft, where our approach tracks closely to whole-model finetuning while still being real-time feasible.

Qualitative Results. Fig. 3 shows our model's adaptation visually. After observing only $0.5s$ of data, our model has adapted to the significant domain shift between nuScenes and Lyft (data frequency mismatch and map format differences). Fig. 3 (right) shows that our proposed sampling scheme yields diverse, multimodal predictions, even in an unusual intersection in Palo Alto.

5 Conclusion

In this work, we present a model that combines the strength of recurrent models with adaptive, optimization-based meta-learning. 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 perform 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.

Acknowledgments and Disclosure of Funding

We thank Nikolai Smolyanskiy and the rest of the NVIDIA AV Prediction team, John Willes, and Brian Ichter for helpful feedback on this paper.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

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

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

[24] 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.

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

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

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

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.

[34] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv:1801.06146*, 2018.

[35] 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.

[36] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[37] 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.

[38] 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.

[39] 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.

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

[41] J. Harrison, *Uncertainty and Efficiency in Adaptive Robot Learning and Control*. PhD thesis, Stanford University, Dept. of Mechanical Engineering, 2021.

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

[43] 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.

[44] 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.

[45] B. Ivanovic and contributors, "trajdata: A unified interface to many trajectory forecasting datasets," 2022. Available at `https://github.com/NVlabs/trajdata`.

A Appendix

A.1 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 [17]), the adaptive formulation allows tasks to vary over time within an episode, similar to meta-learning for continual learning [40]. In particular, our approach builds on an extension to ALPaCA [8] developed in [41]. 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 [42].

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

$$\boldsymbol{y}_t = \Phi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)\boldsymbol{w}_t + \boldsymbol{\varepsilon}_t, \tag{1}$$

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

$$\boldsymbol{w}_{t+1} = A_{\boldsymbol{\theta}}(\boldsymbol{x}_t)\boldsymbol{w}_t + \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) + \boldsymbol{\nu}_t \tag{2}$$

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 [41].

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

$$\begin{aligned}\bar{\boldsymbol{w}}_{t+1|t} &= A_t\bar{\boldsymbol{w}}_{t|t} + \boldsymbol{b}_t\\ S_{t+1|t} &= A_t S_{t|t} A_t + \Sigma_\nu,\end{aligned} \tag{3}$$

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

$$\begin{aligned}P_{t+1} &= \Phi_{t+1} S_{t+1|t}\Phi_{t+1}^\top + \Sigma_\varepsilon\\ K_{t+1} &= S_{t+1|t}\Phi_{t+1}^\top P_{t+1}^{-1}\\ \boldsymbol{e}_{t+1} &= \boldsymbol{y}_{t+1} - \Phi_{t+1}\bar{\boldsymbol{w}}_{t+1|t}\\ \bar{\boldsymbol{w}}_{t+1|t+1} &= \bar{\boldsymbol{w}}_{t+1|t} + K_{t+1}\boldsymbol{e}_{t+1}\\ S_{t+1|t+1} &= S_{t+1|t} - K_{t+1}\Phi_{t+1}S_{t+1|t}.\end{aligned} \tag{4}$$

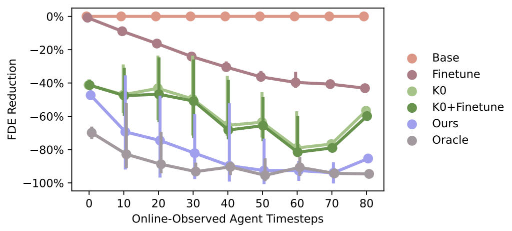
From this framework, the predictive loss is the log likelihood of the prediction using mean and variance $\bar{\boldsymbol{w}}_{t+1|t}, S_{t+1|t}$. Critical to the meta-learning formulation, the parameters of the feature matrix, (possibly) the parameter dynamics matrices, and the noise covariances are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

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 [43].

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

Table 2: Final 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.

	A2B	A2C	A2D	A2E	B2A	B2C	B2D	B2E	C2A	C2B	C2D	C2E	D2A	D2B	D2C	D2E	E2A	E2B	E2C	E2D	AVG
S-STGCNN [28]	3.24	2.86	2.53	2.43	5.16	2.51	4.86	2.88	2.30	1.34	1.74	1.10	2.21	1.99	1.41	0.88	2.10	2.05	1.47	1.01	2.30
PECNet [29]	3.33	2.83	2.53	2.45	5.23	2.48	4.90	2.86	2.22	1.32	1.68	1.12	2.20	2.05	1.52	0.88	2.10	1.84	1.45	0.98	2.29
RSBG [30]	3.42	2.96	2.75	2.50	5.28	2.59	5.19	3.10	2.36	1.55	1.99	1.37	2.28	2.22	1.77	0.97	2.19	2.29	1.81	1.34	2.50
Tra2Tra [31]	3.29	2.88	2.66	2.45	5.22	2.50	4.89	2.90	2.29	1.33	1.78	1.09	2.26	2.12	1.63	0.92	2.18	2.06	1.52	1.17	2.34
SGCN [32]	3.22	2.81	2.52	2.40	5.18	2.47	4.83	2.85	2.24	1.32	1.71	1.03	2.23	1.90	1.48	0.97	2.10	1.95	1.52	0.99	2.29
T-GNN [13]	2.18	2.25	1.78	1.84	4.15	1.82	4.04	2.53	**1.91**	1.12	1.30	0.87	**1.92**	1.46	1.25	**0.65**	**1.86**	1.45	1.28	**0.72**	1.82
K_0	**0.67**	**1.42**	**1.28**	**0.88**	**1.69**	**1.19**	**1.02**	**0.70**	1.97	**0.75**	**0.89**	**0.79**	2.06	**1.04**	**1.18**	1.11	1.87	**0.65**	**1.05**	1.08	**1.16**



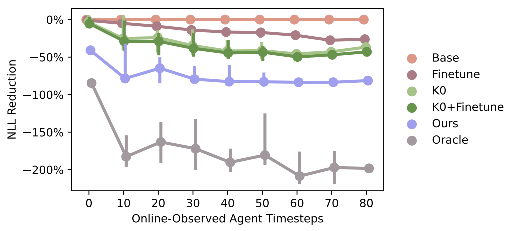


Figure 4: [Zara1 $\rightarrow$ Hotel] Our method's median FDE and NLL reductions replicate the ADE results from Fig. 2, significantly outperforming naïve transfer and other ablations, even matching the oracle on FDE. Error bars are 95% CIs, lower is better.

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 other specialized cores for neural networks, parallelized matrix multiplications are extremely efficient, and thus many samples can be parallelized across.

A.2 Experimental Setup

Our approach was implemented in PyTorch [44] and all datasets were interfaced through `trajdata` [45], a recently-released unified interface to trajectory forecasting datasets.

ETH/UCY Evaluation Protocol. As in recent prior work [13], we treat each of the 5 scenes in the ETH and UCY datasets as individual source domains $\mathcal{S} \in \{$`ETH`, `Hotel`, `Univ`, `Zara1`, and `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 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 [13], 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.

nuScenes $\rightarrow$ Lyft 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.

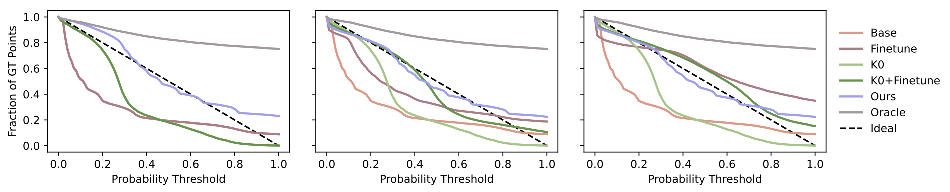


Figure 5: [Zara1 $\rightarrow$ Hotel] Detailed calibration plots showing the fraction of GT future positions lying within specified probability thresholds after observing 0, 1000, and 2000 data samples in the offline setting. Our method is closest to the ideal line, and stays close as more data points are observed. The area below the ideal line signifies underconfidence (e.g., naïve transfer with the Base model). Accordingly, the area above the ideal line signifies overconfidence (especially visible for the oracle). Computing the area between each line to the ideal line yields the ECE values after 0, 1000, and 2000 updates in Fig. 1.

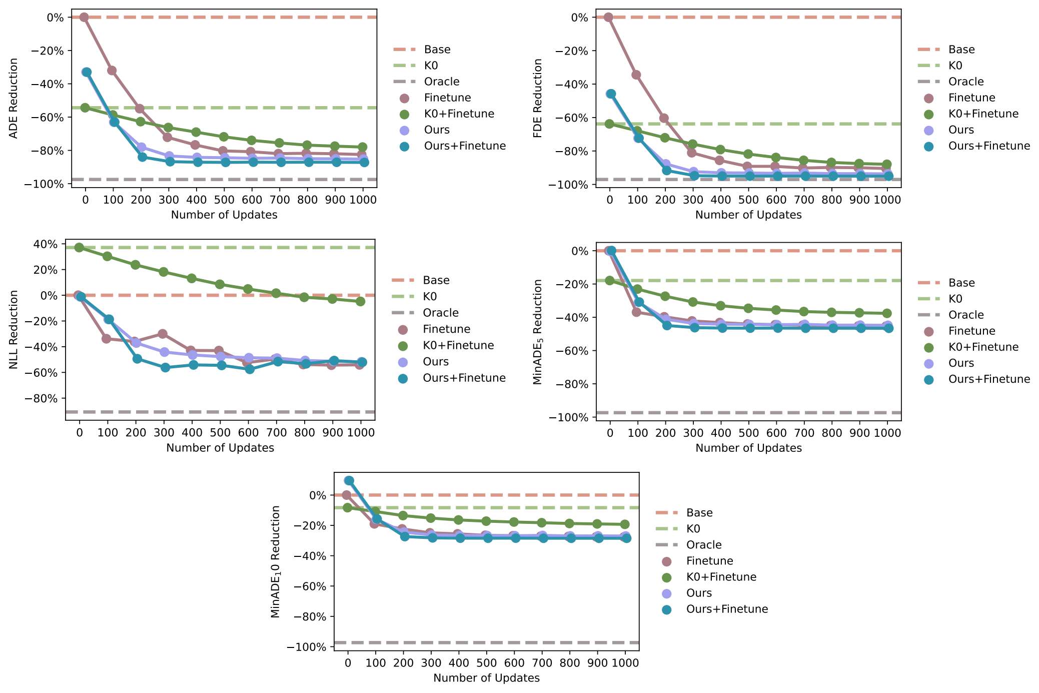


Figure 6: [nuScenes $\rightarrow$ Lyft] Even in the face of significant domain shift (i.e., 5$\times$ timestep frequency and different map annotations), our method's predictions improve smoothly with more data, outperforming whole-model fine-tuning after only 200 data samples (if not immediately). Lower is better.

A.3 Additional Pedestrian Evaluations

Table 2 shows the same analyses as in Table 1, but with the FDE evaluation metric. Fig. 4 shows the same analyses as Fig. 2, but with the FDE and NLL evaluation metrics. Lastly, Fig. 5 shows detailed calibration plots which underlie the ECE computations in Fig. 1. Broadly, the results of these additional evaluations confirm and reinforce the performance of our approach over other baselines and ablations.

A.4 Additional Autonomous Driving Evaluations

As can be seen in Fig. 6, 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.

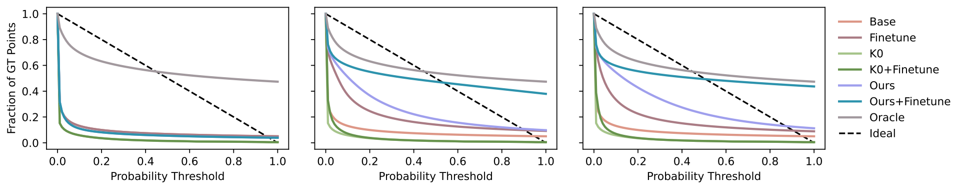


Figure 7: [nuScenes → Lyft] Detailed calibration plots showing the fraction of GT future positions lying within specified probability thresholds after observing 0, 500, and 1000 data samples in the offline setting. Our method fast approaches the ideal line (with Ours+Finetune doing so even faster), and stays better calibrated than only performing whole-model finetuning (Finetune). The area below the ideal line signifies underconfidence (e.g., naïve transfer with the Base model). Accordingly, the area above the ideal line signifies overconfidence. Computing the area between each line to the ideal line yields the ECE values after 0, 500, and 1000 updates in Fig. 1.

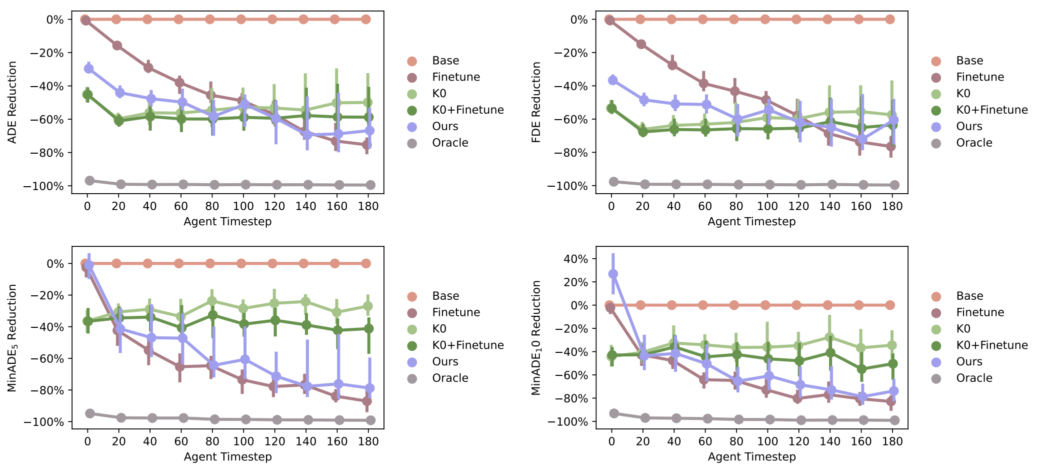


Figure 8: [nuScenes → Lyft] Evaluating our approach on additional metrics matches the result in Fig. 2; our approach rapidly improves online as it observes more data. Error bars are 95% CIs, lower is better.

Fig. 7 shows detailed calibration plots which underlie the ECE computations in Fig. 1. Finally, Fig. 8 shows the same analyses as in Fig. 2, but on the other four evaluation metrics. Overall, the results of these additional evaluations confirm and reinforce the performance of our approach over other baselines and ablations.