Robust Forecasting for Robotic Control: A Game-Theoretic Approach

Shubhankar Agarwal
University of Texas at Austin
`somi.agarawl@utexas.edu`

David Fridovich-Keil
University of Texas at Austin
`dfk@utexas.edu`

Sandeep P. Chinchali
University of Texas at Austin
`sandeepc@utexas.edu`

Abstract

Modern robots require accurate forecasts to make optimal decisions in the real world. For example, self-driving cars need an accurate forecast of other agents' future actions to plan safe trajectories. Current methods rely heavily on historical time series to accurately predict the future. However, relying entirely on the observed history is problematic since it could be corrupted by noise, have outliers, or not completely represent all possible outcomes. We propose a novel framework for generating robust forecasts for robotic control to solve this problem. To model real-world factors affecting future forecasts, we introduce the notion of an adversary, which perturbs observed historical time series to increase a robot's ultimate control cost. Specifically, we model this interaction as a zero-sum two-player game between a robot's forecaster and this hypothetical adversary. We show that our proposed game may be solved to a local Nash equilibrium using gradient-based optimization techniques. Furthermore, a forecaster trained with our method performs 30.14% better on out-of-distribution real-world lane change data than baselines.

1 Introduction

Robots deployed in the real world rely on accurate forecasts of the future to make reliable decisions amidst uncertainty. For example, an autonomous vehicle must forecast the trajectory of cars in an adjacent lane in order to decide when and how to change lanes. Current methods such as [1]–[3] rely heavily on historical time series to accurately predict the future. However, completely relying on the observed history is problematic since it could be corrupted by sensor noise, have outliers, or not completely represent all possible outcomes. Current practices, such as collecting more targeted data or adding random noise to existing data, are expensive or not reflective of important outliers.

We propose a novel framework for generating robust forecasts for optimal decision-making in robotics. In our system, the forecaster observes historical time series data and makes a prediction which a controller uses to determine optimal future actions. To model real-world factors affecting future forecasts, we introduce a notion of an adversary, which perturbs the historical time series, leading to inaccurate forecasts. As such, the forecaster and adversary play a game where the adversary tries to maximize its reward by perturbing the forecaster's input, while the forecaster minimizes its cost by performing well on the adversarially perturbed inputs. Motivated by this observation, our contributions are as follows. **Contributions:** First, we formulate the problem of robust forecasting for optimal control as a two-player, zero-sum game and show it reaches local Nash equilibrium (LNE). Second, we show that our robustly-trained forecaster achieves a 30.14% better performance on out-of-distribution real-world lane change data than baseline forecasters.

NeurIPS 2022 Workshop on Robot Learning: Trustworthy Robotics, Virtual, Virtual

Related works can be categorized into two main sub-categories. 1) Forecasting Models: [4]–[13]. These works either do not account for outliers in the data or consider robustness to adversarial distribution shifts. 2) Robust Optimization and adversarial training: [14]–[20]. Most of these works focus on robust control or vision models; however, they do not consider the downstream implications on control performance for robotic decision problems. We discuss related works in detail in Section 4.1.

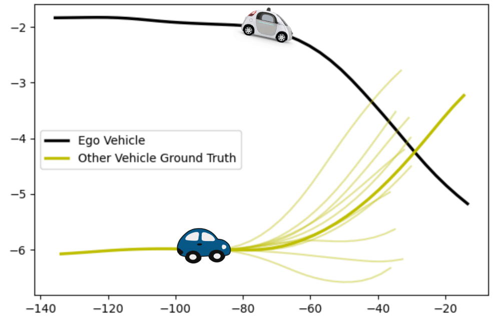

Figure 1: **Motivating Example:** The ego vehicle (white) needs to plan its future trajectory given historical observations of the other vehicle (bold yellow line). Other highlighted yellow lines show several possible trajectories for the blue car, some resulting in faster or even unsuccessful lane changes. Even though the ego vehicle forecaster observes only a single scenario (bold yellow line), it needs to be robust to other possible outcomes.

2 Problem and Approach

Forecaster: The forecaster, $f : \mathbb{R}^{p\times H} \rightarrow \mathbb{R}^{p\times F}$, maps the time series of past H measurements, denoted by s_H, to a time series of future F measurements, denoted by $\hat{s}_F$. The hat notation, $\hat{s}$, denotes predicted values of the forecaster, and s_F denotes the ground-truth time series. The forecaster is a learned module parameterized by $\theta_f \in \Theta_f$. To simplify notation, we use bold variables to define the full-time series, i.e., the time series of past H measurements as $\mathbf{s_H}$.

Controller: The control policy $\pi : \mathbb{R}^n \times \mathbb{R}^{p\times F} \rightarrow \mathbb{R}^m$ maps the state of system, $x_t \in \mathbb{R}^n$, and time series of future F measurements, $\mathbf{s_F}$, to an optimal control $u_t \in \mathbb{R}^m$. We denote the state and control constraint sets by $\mathcal{X}$ and $\mathcal{U}$, respectively. The robot dynamics, g, are given by: $x_{t+1} = g(x_t, u_t), \forall t \in \{0, \ldots, T-1\}$. In practice, given a possibly perturbed forecast $\hat{\mathbf{s}}_\mathbf{F}$, it will enact a control denoted by $\hat{u}_t = \pi(x_t, \hat{\mathbf{s}}_\mathbf{F}; \theta_c)$, which depends on the forecaster parameters θ_f via the forecast $\hat{\mathbf{s}}_\mathbf{F}$. The main objective is to minimize the control cost J^C, which depends on initial state x_0 and controls $\hat{u}_{0:T-1}$. The control cost J^C is a sum of stage costs $c(x_t, s_t, \hat{u}_t)$ and terminal cost $c_T(x_T, s_T)$, i.e., $J^C(\hat{\mathbf{u}}; x_0, \mathbf{s}) = \sum_{t=0}^{T-1} c(x_t, s_t, \hat{u}_t) + c_T(x_T, s_T)$.

Adversary: To account for the measurement noise in the inputs of the forecaster and improve generalization to out-of-distribution data, we introduce an adversary. The adversary is defined as the map $a : \mathbb{R}^{p\times H} \rightarrow \mathbb{R}^{p\times H}$, which takes as input the time series of past H measurements, $\mathbf{s_H}$, and outputs an adversarially perturbed version of the past H measurements denoted as $\mathbf{s}_\mathbf{H}^{\text{adv}}$. The adversary is parameterized by $\theta_a \in \Theta_a$. To restrict the adversary's power, we penalize the adversary quadratically for large perturbations: $\|\mathbf{s_H} - \mathbf{s}_\mathbf{H}^{\text{adv}}\|_2^2$.

Overall Cost: First, the adversary (with parameters θ_a) perturbs the historical time series, $\mathbf{s}_\mathbf{H}^{\text{adv}}$, given the actual history of time series $\mathbf{s_H}$, $\mathbf{s}_\mathbf{H}^{\text{adv}} = a(\mathbf{s_H}; \theta_a)$. Then, the forecaster observes the adversary's perturbed history, $\mathbf{s}_\mathbf{H}^{\text{adv}}$, and predicts the system's future state: $\hat{\mathbf{s}}_\mathbf{F} = f(\mathbf{s}_\mathbf{H}^{\text{adv}}; \theta_f)$. Finally, given the predicted forecast $\hat{\mathbf{s}}_\mathbf{F}$ we calculate the corresponding optimal controls: $\hat{\mathbf{u}} = \pi\left(x_0, \hat{\mathbf{s}}_\mathbf{F}; \theta_c\right)$. Similarly, we calculate the optimal controls: $\mathbf{u} = \pi\left(x_0, \mathbf{s_F}; \theta_c\right)$, for the ground-truth forecast $\mathbf{s_F}$.

Thus equipped, we calculate the overall cost (1). The first term calculates the additional cost incurred by using predicted forecasts $\hat{\mathbf{s}}_\mathbf{F}$ instead of true forecast $\mathbf{s_F}$. This term models the change in states x and controls u given the errors in predicting the future time series. The second term $\|\mathbf{s_F} - \hat{\mathbf{s}}_\mathbf{F}\|_2^2$ penalizes deviations of the forecaster's future time series predictions and the ground-truth forecast. The third term $\|\mathbf{s_H} - \mathbf{s}_\mathbf{H}^{\text{adv}}\|_2^2$ controls the adversary's power by penalizing it for making large perturbations. The hyper-parameters λ_f and λ_a control the relative importance of the respective costs:

$$J\left(\cdot\right) = J^C\left(\hat{\mathbf{u}}; x_0, \mathbf{s_F}\right) - J^C\left(\mathbf{u}; x_0, \mathbf{s_F}\right) + \lambda_f\|\mathbf{s_F} - \hat{\mathbf{s}}_\mathbf{F}\|_2^2 - \lambda_a\|\mathbf{s_H} - \mathbf{s}_\mathbf{H}^{\text{adv}}\|_2^2. \tag{1}$$

For clarity, we introduce the following compact notation for this cost—$\mathcal{J}(\theta_f, \theta_a)$ is the final overall cost $J\left(\mathbf{u}, \hat{\mathbf{u}}, \mathbf{s_F}, \hat{\mathbf{s}}_\mathbf{F}; x_0\right)$ with fixed parameters θ_f and θ_a. The total cost depends on the controller and forecaster parameters via controls $\mathbf{u}$ and $\hat{\mathbf{u}}$ and the forecast $\mathbf{s_F}$. We now formalize the problem.

Problem 1 (Adversarially-Robust Control)**.** *Given a forecaster f and an adversary a, we aim to find a saddle point (min and max order does not matter) of the following problem:*

$$\min_{\theta_f} \max_{\theta_a} \quad \mathcal{J}(\theta_f, \theta_a) \tag{2a}$$

$$\text{subject to : } \mathbf{s}_{\mathbf{H}}^{adv} = a(\mathbf{s}_{\mathbf{H}}; \theta_a),\ \hat{\mathbf{s}}_{\mathbf{F}} = f(\mathbf{s}_{\mathbf{H}}^{adv}; \theta_f), \tag{2b}$$

$$\hat{\mathbf{u}} = \pi\left(x_0, \hat{\mathbf{s}}_{\mathbf{F}}; \theta_c\right),\ \mathbf{u} = \pi\left(x_0, \mathbf{s}_{\mathbf{F}}; \theta_c\right). \tag{2c}$$

In Problem 1, the adversary is trying to maximize the overall control cost (2a) by perturbing the original past H measurements, $\mathbf{s}_{\mathbf{H}}$ (2b). In contrast, the forecaster is trying to minimize the overall cost, (2a), by forecasting the future F measurements, $\hat{\mathbf{s}}_{\mathbf{F}}$ (2b). Intuitively, this problem captures how to find a forecaster that is robust to adversarial perturbations for the purposes of reliable robotic decision-making. All components operate in discrete time steps t for a horizon of T steps.

2.1 Robust Forecasting Game Characterization

We observe that Problem 1 is a two-player, zero-sum game, and seek both forecaster and adversary parameters that are in equilibrium. We characterize the *Robust Forecasting Game* as follows: **Player 1 (Forecaster):** The forecaster's goal is to predict relevant future system states, despite perturbations of the history by the adversary. In the lane changing example of Fig. 1, this future state is the upcoming trajectory of the blue car. As such, the forecaster seeks parameters θ_f^* which minimize the overall cost in Problem 1 despite worst-case adversarial parameter selection, θ_a^*. **Player 2 (Adversary):** The adversary's goal is to provide a perturbed history to the forecaster such that the predicted future time series by the forecaster incurs a higher overall cost. In particular, for any fixed choice of forecaster parameter θ_f, it seeks parameters θ_a which maximize the overall cost in Problem 1, $\mathcal{J}(\cdot, \theta_f)$.

A global Nash equilibrium (GNE) [14, Def. 2.1] is a point in the space of game strategies where both players cannot change their strategies without achieving a less favorable outcome. A LNE [21, Def. 1] is a point in the space of strategies where this property need only hold within a small neighborhood. LNEs are characterized by first- (Prop. 1) and second-order (Prop. 2) optimality conditions.

Training the Models: In this work, we use feedforward neural networks to represent the forecaster and adversary models. Due to the nonconvexities in overall cost $\mathcal{J}$, we can at best guarantee that our proposed game will reach a LNE. More precisely, [22] demonstrates that stochastic gradient descent methods do not necessarily converge to LNE in zero-sum games, but [23] proposes a new second-order gradient update rule that does guarantee convergence to a LNE if one exists. However, due to the complexities and speed limitations of the second-order gradient update method, we follow standard practices in high-dimensional optimization and resort to an adaptive first-order method such as ADAM [24]. Since we do not use the theoretically-motivated second-order method of [23], we check the first- and second-order conditions of Propositions 1 and 2 to ensure that we have found a LNE. The forecaster and adversary are trained via alternating gradient steps using the whole training dataset. We repeat this process until convergence, and subsequently, check the necessary and sufficient conditions of LNE to check if the converged parameters θ_f^* and θ_a^* constitute a LNE.

3 Experiments

We now evaluate our method on lane-change data from an autonomous driving scenario with human participants [5]. The experiment aims to demonstrate the forecaster trained using our proposed method will be robust to Out-of-Distribution (OoD) data. *Models:* Both the forecaster and adversary are NN models with two fully connected layers and ReLU activations. *Differentiable Model Predictive Control (MPC):* In our experiment, the control policy π is the solution map of a differentiable MPC problem with quadratic costs and linear constraints, discussed in detail in Section 4.3.1.

Datasets and Benchmark Algorithms: We compare various forecasters trained on the following datasets, which we call training schemes: 1) ORIGINAL: The forecaster is *only* trained on $\mathcal{D}_{\text{orig}}^{\text{train}}$. 2) DATA ADDED: We add more training examples from the same distribution as the original training dataset, denoted by $\mathcal{D}_{\text{add}}^{\text{train}}$. This tests whether more examples can improve performance. 3) RANDOM: We apply zero-mean Gaussian noise with unit variance at each time step to the original training data. The perturbed dataset is denoted by $\mathcal{D}_{\text{rand}}^{\text{train}}$, and we re-train the task model on $\mathcal{D}_{\text{rand}}^{\text{train}} \cup \mathcal{D}_{\text{orig}}^{\text{train}}$. 4) ROBUST (Ours): We use our proposed method to train the forecaster. For a fair comparison, the DATA ADDED, RANDOM, and ROBUST schemes add the same number of new training examples to the original training dataset. The datasets are further defined in detail in Section 4.3.2.

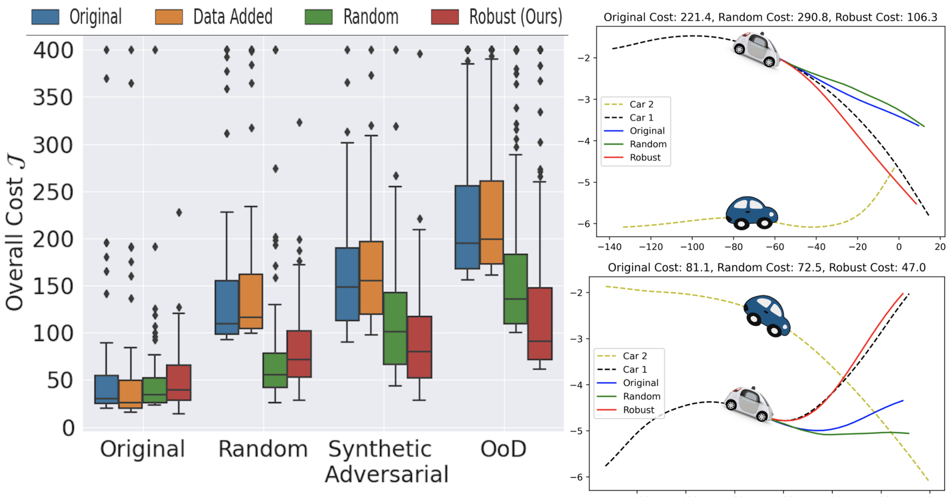

Figure 2: **Benefits of game-based training**

Lane-Change Forecasting: In our experiment Fig. 2, the forecaster's (a NN) goal was to predict the ego vehicle's (white car) future trajectory to complete a successful lane change, given the history of states of both cars (blue and white). To model the measurement noise and uncertainty around the other car's decision-making, we restricted the adversary (a NN) to only be able to perturb the other car's (blue) observed historical time series. Exact implementation details are described in Section 4.3.3. The training dataset for all schemes contained low-speed lane change scenarios ($\leq 35\,\mathrm{m\,s^{-1}}$). In order to evaluate the generalization capabilities of the training schemes, we compared the training schemes on a OoD test dataset consisting of high-speed lane change scenarios ($\geq 35\,\mathrm{m\,s^{-1}}$). The OoD dataset represents scenarios not seen in the training but which are still possible in the real world, and therefore our forecaster should be robust to them.

Results: Figure 2 shows the overall cost for all schemes on *test* datasets. The key takeaways from our experiments are as follows: 1) **All forecasters perform similarly on the original test data.** The DATA ADDED (orange)scheme's best performance is expected because it is trained on more original data than the other schemes. 2) **Our trained adversary generates challenging scenarios.** We run the final trained adversary with final parameters θ_a^* on the held-out original test dataset to generate unseen adversarial scenarios, which form the *synthetic adversarial* test dataset. The poor performance of all training schemes on the synthetic adversarial test dataset confirms that the adversary has learned perturbations that are hard for the forecaster, leading to higher control costs. However, our ROBUST scheme (red) performs significantly better since it was trained to anticipate such unseen perturbations. 3) **Our ROBUST scheme performs 30.14% better compares to RANDOM on naturally occurring OoD test dataset**. The results show that our ROBUST training scheme can learn robustness to adversarial and OoD scenarios. The RANDOM scheme can generalize better to OoD data but cannot match our ROBUST scheme's performance since the random perturbations are relatively benign compared to the targeted scenarios generated by our algorithm.

Qualitative Results: In Fig. 2 (right), we demonstrate our method's performance on two OoD lane change scenarios. Specifically, we show trajectories that are forecasted by models trained using the ORIGINAL (blue), RANDOM (green), and ROBUST (red) training schemes given the historical time series. None of the training schemes were exposed to this OoD scenarios at train time. The ground-truth trajectories are shown as dotted lines. All the forecasters were given the same historical time series (time series before the car locations) of both cars to predict the future time series of the ego vehicle. The control cost J^C of each forecasted trajectory is shown in the legend. While the ORIGINAL and RANDOM training schemes cannot correctly predict the ego-vehicle future trajectory, our ROBUST training scheme can correctly predict the ego-vehicle future trajectory and has the lowest control cost. **The performance of our ROBUST training scheme highlights how our game formulation helps the forecaster generalize better to OoD scenarios.**

Future Work: In our experiments, we considered a simpler case where we did open-loop planning for both agents in the lane-change scenarios. In future work, we will model each agent's outputs as a distribution in order to explicitly account for multiple possible outcomes and formulate the problem as a stochastic partial information dynamic game [14, Sec. 6.4].

References

[1] B. Lim, “Deep learning for time series prediction and decision making over time,” University of Oxford, Tech. Rep., 2021.

[2] B. Ivanovic, “Trajectory forecasting in the modern robotic autonomy stack,” Standford University, Tech. Rep., 2021.

[3] S. Makridakis, “A survey of time series,” *International Statistical Review / Revue Internationale de Statistique*, vol. 44, no. 1, pp. 29–70, 1976, ISSN: 03067734, 17515823.

[4] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, “Context-aware trajectory prediction,” *arXiv preprint arXiv:1705.02503*, 2017.

[5] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, “Multimodal probabilistic model-based planning for human-robot interaction,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia: IEEE Press, 2018, pp. 1–9.

[6] A. Alahi, K. Goel, V. Ramanathan, *et al.*, “Social LSTM: Human trajectory prediction in crowded spaces,” in *Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[7] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, “Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection,” *Neural networks*, vol. 108, pp. 466–478, 2018.

[8] N. Lee, W. Choi, P. Vernaza, *et al.*, “DESIRE: Distant future prediction in dynamic scenes with interacting agents,” in *Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 336–345.

[9] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, *Transformer networks for trajectory forecasting*, 2020. arXiv: `2003.08111 [cs.CV]`.

[10] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, “Precog: Prediction conditioned on goals in visual multi-agent settings,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[11] B. Varadarajan, A. Hefny, A. Srivastava, *et al.*, “Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction,” in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7814–7821.

[12] H. Song, D. Luan, W. Ding, *et al.*, “Learning to predict vehicle trajectories with model-based planning,” in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 1035–1045.

[13] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data,” in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII*, Glasgow, United Kingdom: Springer-Verlag, 2020, pp. 683–700, ISBN: 978-3-030-58522-8.

[14] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics, 1998. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611971132`.

[15] P. M. Esfahani and D. Kuhn, *Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations*, 2015.

[16] A. Sinha, H. Namkoong, and J. Duchi, “Certifiable distributional robustness with principled adversarial training,” in *International Conference on Learning Representations*, 2018.

[17] R. Volpi, H. Namkoong, O. Sener, *et al.*, “Generalizing to unseen domains via adversarial data augmentation,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, *et al.*, Eds., vol. 31, Curran Associates, Inc., 2018.

[18] A. Madry, A. Makelov, L. Schmidt, *et al.*, “Towards deep learning models resistant to adversarial attacks,” in *International Conference on Learning Representations*, 2018.

[19] E. Wong and Z. Kolter, “Provable defenses against adversarial examples via the convex outer adversarial polytope,” in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 5286–5295.

[20] A. Ilyas, S. Santurkar, D. Tsipras, *et al.*, “Adversarial examples are not bugs, they are features,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, *et al.*, Eds., vol. 32, Curran Associates, Inc., 2019.

[21] L. Ratliff, S. Burden, and S. Sastry, "On the characterization of local Nash equilibria in," *IEEE Transactions on Automatic Control*, vol. 61, pp. 1–1, Aug. 2016.

[22] E. Mazumdar, L. J. Ratliff, and S. S. Sastry, "On gradient-based learning in continuous games," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 1, pp. 103–131, 2020.

[23] E. V. Mazumdar, M. I. Jordan, and S. S. Sastry, "On finding local Nash equilibria (and only local Nash equilibria) in zero-sum games," *ArXiv*, vol. abs/1901.00838, 2019.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[27] U. Ghai, D. Snyder, A. Majumdar, and E. Hazan, "Generating adversarial disturbances for controller verification," in *Learning for Dynamics and Control*, PMLR, 2021, pp. 1192–1204.

[28] N. Agarwal, B. Bullins, E. Hazan, *et al.*, "Online control with adversarial disturbances," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 111–119.

[29] P.-h. Li, U. Topcu, and S. P. Chinchali, *Adversarial examples for model-based control: A sensitivity analysis*.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, *et al.*, Eds., vol. 27, Curran Associates, Inc., 2014.

[31] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *International Conference on Learning Representations*, 2017.

[32] A. U. Raghunathan, A. Cherian, and D. K. Jha, *Game theoretic optimization via gradient-based nikaido-isoda function*, 2019.

[33] T. Fiez, B. Chasnov, and L. J. Ratliff, "Convergence of learning dynamics in Stackelberg games," *arXiv preprint arXiv:1906.01217*, 2019.

[34] T. Fiez, L. Ratliff, E. Mazumdar, *et al.*, "Global convergence to local minmax equilibrium in classes of nonconvex zero-sum games," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 049–29 063, 2021.

[35] L. Zheng, T. Fiez, Z. Alumbaugh, *et al.*, "Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 9217–9224.

[36] F. Laine, D. Fridovich-Keil, C.-Y. Chiu, and C. Tomlin, "The computation of approximate generalized feedback Nash equilibria," *arXiv preprint arXiv:2101.02900*, 2021.

[37] A. Agrawal, B. Amos, S. Barratt, *et al.*, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, 2019.

[38] V. S. GmbH, *Vtd - virtual test drive*.

4 Appendix

4.1 Related Works

Forecasting: Forecasting ego-vehicle trajectories is primarily studied via data-driven and probabilistic techniques. Data-driven trajectory forecasting approaches, such as [4], [6]–[9], treat both single and multi-trajectory forecasting problems within a framework of temporal regression, using models such as Long-Short Term Memories (LSTMs) [25] and Transformers [26]. These methods typically do not account for outliers in the dataset or OoD scenarios. In contrast, probabilistic forecasting models such as [5], [10]–[13] output a distribution of possible future trajectories, and can thus account for outliers and multimodality. Still, these techniques have difficulty generalizing to OoD scenarios since the distributions they learn are based only upon fixed training data. In particular, none of these approaches consider robustness to adversarial distribution shifts.

Adversarial Machine Learning for Control: Adversarial modeling is widely studied in the context of robust control and decision making [14]–[17]. For example, the introduction of an adversary to improve a machine learning model's robustness and generalization capabilities is commonly studied in vision and classification tasks such as [18]–[20]. However, existing works in robust machine learning do not consider the downstream implications on control performance for robotic decision problems. Works which do study adversarial attacks in control systems consider settings in which control decisions are directly perturbed by adversarial inputs [27], [28]. They do not consider a more general setting in which inputs are provided to a forecaster which then invokes an internal model-based controller, as we consider in this work. Additionally, [29] design adversarial attacks which increase a cost metric as well as violate state and control constraints. However, they did not study how to exploit these adversarial attacks to design an improved, robust decision process for robotic applications with a forecaster and a model-based controller.

Game Theory: Learning in noncooperative settings such as games exposes significant challenges regarding convergence, stability of desired solutions, etc. For example, these problems are well known in the context of generative adversarial networks [30], which are notoriously difficult to train [31]. However, recent advances in numerical game theory [22], [23], [32]–[36] provide important steps forward on several of these fronts. In particular, these algorithmic advances ensure that local equilibrium solutions to the games considered in this work can be found reliably.

4.2 Preliminaries

Proposition 1 (First-order Necessary Condition)**.** *[21] Assuming $\mathcal{J}$ is differentiable, any local Nash equilibrium satisfies* $\nabla_{\theta_f}\mathcal{J}(\theta_f, \theta_a) = 0$ *and* $\nabla_{\theta_a}\mathcal{J}(\theta_f, \theta_a) = 0$.

Proposition 2 (Second-order Sufficient Condition)**.** *[21] Assuming $\mathcal{J}$ is twice-differentiable, any local Nash equilibrium satisfies* $\nabla^2_{\theta_f\theta_f}\mathcal{J}(\theta_f, \theta_a) \succeq 0$, *and* $\nabla^2_{\theta_a\theta_a}\mathcal{J}(\theta_f, \theta_a) \preceq 0$.

4.3 Experiments

4.3.1 Differentiable Model Predictive Control (MPC)

In our experiment, the control policy π is the solution map of an MPC problem with quadratic costs and linear constraints. The forecaster provides the MPC controller with a future time series forecast $\hat{\mathbf{s}}_{\mathbf{F}}$ to track and the current state, x_0. We use linear dynamics, g, in our MPC formulation. Specifically, we used second-order linear dynamics for the lane-change experiment. The stage cost $c(x_t, s_t, \hat{u}_t)$ and terminal cost $c_T(x_T, s_T)$ in the control cost J^C are quadratic in the state x_T and controls $\hat{u}_T$. Specifically, the terminal cost is $c_T(x_T, s_T) = (x_T - s_T)^\top Q\,(x_T - s_T)$ and the stage cost is $c(x_t, s_t, \hat{u}_t) = (x_t - s_t)^\top Q\,(x_t - s_t) + \hat{u}_t{}^\top R\hat{u}_t$, where Q and R are positive definite matrices. The robot actuator constraints which are described by intervals along each axis, i.e. $u_{\text{min}} \le u_t \le u_{\text{max}}$. Likewise, we presume that states are also constrained to lie within an axis-aligned box: $x_{\text{min}} \le x_t \le x_{\text{max}}$. While training the forecaster and the adversary, we require gradients of the control policy with respect to the forecaster and the adversary parameters. To do so, we use the CVXPYLAYERS Pytorch library [37], which allows us to backpropagate derivative information through convex optimization problems and thereby train both the forecaster and adversary end-to-end.

4.3.2 Dataset

In the experiment, the time series forecasts are a tensor instead of a vector. For example, the historical time series is $\mathbf{s_H} \in \mathbb{R}^{N \times p \times H}$, where N is the number of individual time series, p is the dimension, and H is the horizon of the time series. We collect several examples of these time series tensors in a dataset, which we use to train the forecaster and the adversary. A dataset contains N tuples of inputs x and labels y denoted by $\mathcal{D} = \{(x, y)\}_{i=1}^{N}$. In each tuple, $x = \{\mathbf{s_H}, x_0\}$ and $y = (\mathbf{s_F})$. From these, the forecaster predicts a future time series $\hat{y} \equiv \hat{\mathbf{s}}_\mathbf{F}$. The subscript b in the dataset $\mathcal{D}_b^a$ indicates the type of dataset, such as whether it is original or adversarially generated. Likewise, the superscript a indicates if the dataset is from the *train* or *test* distribution.

4.3.3 Lane Change Forecaster

Now, we demonstrate our method's real-world applicability on a challenging lane change dataset [38] used to train self-driving policies. This dataset contains 1105 human-human interactive lane change trials from over 19 volunteer drivers in a driving simulator. The drivers had to swap lanes with each other within $135\,\mathrm{m}$ of a straight road. The state of each vehicle is $x_t = [p_x, p_y, v_x, v_y] \in \mathbb{R}^4$, where p_x and p_y are the 2-D position of the car in meters and v_x and v_y are the 2-D velocity of the car in $\mathrm{m\,s^{-1}}$. The control variable is $u_t = [a_x, a_y] \in \mathbb{R}^2$, where a_x and a_y are the 2-D acceleration of the car in $\mathrm{m\,s^{-2}}$. Each scenario begins with initial conditions drawn randomly. Our training dataset is of size $N = 500$, and the test dataset is of size $N = 100$.

The forecaster's goal was to predict the ego vehicle's future trajectory to complete a successful lane change, given the history of states of both cars. The historical time series is of horizon $H = 20$ and the future time series of horizon $F = 20$. As such, the forecaster's history time series tensor is $\mathbf{s_H} \in \mathbb{R}^{500 \times 8 \times 20}$, since it contains the history of the time series of both cars. The forecaster's future time series tensor is $\mathbf{s_F} \in \mathbb{R}^{500 \times 4 \times 20}$. To model the measurement noise and uncertainty around the other car's decision-making, we restricted the adversary to only be able to perturb the other car's observed historical time series. Therefore, the adversary took the other car's historical state trajectory as input and generated an adversarially perturbed history for that car. The adversary's historical time series tensor is $\mathbf{s_H} \in \mathbb{R}^{500 \times 4 \times 20}$, since it contains the historical time series of only the other car. For the train and test datasets, we used state trajectories with $v_x, v_y \leq 35\,\mathrm{m\,s^{-1}}$. The control policy π tracks the predicted future trajectory from the forecaster with a quadratic cost function. Additionally, $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}$ matrices in g follow second-order linear dynamics and $Q \in \mathbb{R}^{n \times n}, R \in \mathbb{R}^{m \times m}$ matrices in the state cost are identity matrices. For the OoD dataset $\mathcal{D}_{\mathrm{ood}}^{\mathrm{test}}$, we used real state trajectories with $v_x, v_y > 35\,\mathrm{m\,s^{-1}}$. The OoD dataset represents scenarios not seen in the training distribution but is still possible in the real world, and therefore our forecaster should be robust to them. Both hyperparameters λ_f and λ_a were set to 10.0 and were chosen experimentally to balance forecaster performance with control performance while also allowing significant adversarial perturbations.

Experiment Procedure In our experiment, the forecaster is initialized with pre-trained parameters on the original train dataset using the mean-squared error loss between the predicted forecasts, $\hat{\mathbf{s}}_\mathbf{F}$, and the ground-truth forecasts, $\mathbf{s_F}$. In both experiments, the forecaster and adversary are trained via alternating gradient steps, using the whole training dataset. As such, first, the forecaster makes a prediction from the perturbed history generated from the adversary and updates its parameters θ_f. Then, the adversary predicts the new perturbed history, uses the updated forecaster parameters to calculate the overall cost $\mathcal{J}$ and updates its parameters θ_a. We repeat this process until convergence, and subsequently check the necessary and sufficient conditions of LNE given in Propositions 1 and 2 to check if the converged parameters θ_f^* and θ_a^* constitute a LNE.