2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Robust Forecasting for Robotic Control: A Game-Theoretic Approach

Shubhankar Agarwal[1], David Fridovich-Keil[2], and Sandeep P. Chinchali[1]

***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. To solve this problem, we propose a novel framework for generating robust forecasts for robotic control. In order 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, we show that a forecaster trained with our method performs 30.14% better on out-of-distribution real-world lane change data than baselines.**

I. 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. The problem of accurate forecasting is likewise essential in other large-scale and safety-critical systems, such as electric grids and communications networks, and transportation systems.

However, forecasting the future is one of the most challenging problems in machine learning. 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. For example, in the case of self-driving cars, the observed history could be corrupted by noise due to sensor imprecision or outliers due to human labeling errors. Moreover, the observed historical training dataset might be incomplete—in the case of lane-change maneuvers, we might only observe that if one car slows down, the other car completes the lane change first (see Fig. 1). However, we might not observe outlier behavior where a car slows down and subsequently speeds up. Systems deployed in the real world need to be robust to these problems and generate forecasts which consider these situations. Standard data engineering practices, such as collecting more targeted data or adding random noise to existing data, are either expensive or not reflective of important outliers, respectively.

[1]Departments of Electrical and Computer Engineering and [2] Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin, {somi.agarwal, dfk, sandeepc}@utexas.edu

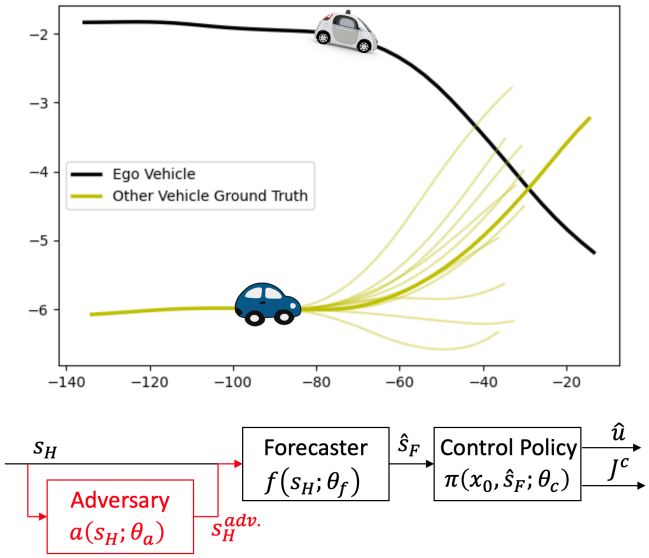

Fig. 1: **Motivating Example (top):** Lane change scenario involving an autonomous "ego" car (white) and another, potentially human-operated car (blue). The ego vehicle needs to plan its future trajectory given historical observations of the other vehicle (bold yellow line). Other highlighted yellow 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 these other possible outcomes. **Architecture (bottom):** Our robotic system consists of a forecaster, hypothetical adversary, and controller.

We propose a novel framework for generating robust forecasts for optimal decision-making in robotics. We consider a system, consisting of a forecaster and a controller, represented in Fig. 1. The forecaster observes historical time series data and makes a prediction which the controller uses to determine optimal future actions. For example, a self-driving car's forecaster predicts the future behavior of a human-driven car and the controller subsequently determines the self-driving car's optimal lane change maneuver in response. To model real-world factors affecting the 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. Recent advances in numerical game theory provide

gradient-based algorithms which are guaranteed to find local Nash equilibriums (LNEs) in such problems. Second, we show the benefits of a robustly-trained forecaster on two different datasets, including a 30.14% improvement in performance on out-of-distribution real-world lane change data as compared to baseline forecasters.

II. Related Work

Forecasting: Forecasting ego-vehicle trajectories is primarily studied via data-driven and probabilistic techniques. Data-driven trajectory forecasting approaches, such as [4]–[8], treat both single and multi-trajectory forecasting problems within a framework of temporal regression, using models such as Long-Short Term Memories (LSTMs) [9] and Transformers [10]. These methods typically do not account for outliers in the dataset or Out-of-Distribution (OoD) scenarios. In contrast, probabilistic forecasting models such as [11]–[15] 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 [16]–[19]. 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 [20]–[22]. 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 [23], [24]. 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, [25] 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. Recent work, [26] employs a similar problem as our method but does not consider the game formulation and thus is not able to provide theoretical insights on convergence or robustification abilities.

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 [27], which are notoriously difficult to train [28]. However, recent advances in numerical game theory [29]–[35] 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.

III. Problem Formulation

We now describe the interplay between the robot's forecaster, controller, and the adversary (see Fig. 1). They all operate in discrete time steps t for a horizon of T steps.

Forecaster: The forecaster observes a time series $s_T \in \mathbb{R}^{p\times T}$, drawn from a data distribution $\mathcal{D}$, denoted by $s_T \sim \mathcal{D}$. The forecaster, $f : \mathbb{R}^{p\times H} \to \mathbb{R}^{p\times F}$, 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 $\theta_f \in \Theta_f$, where Θ_f is the space of all possible parameters. To simplify notation, we use bold variables to define the full time series, i.e., the time series of past H measurements as $\mathbf{s_H}$.

Control Policy: The control policy $\pi : \mathbb{R}^n \times \mathbb{R}^{p\times F} \to \mathbb{R}^m$ 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: $x_{t+1} = g(x_t, u_t), \forall t \in \{0, \dots, T-1\}$. Ideally, control policy π chooses a decision u_t at time t based on fully-observed state x_t and perfect knowledge of exogenous input $\mathbf{s_F}$: $u_t = \pi(x_t, \mathbf{s_F}; \theta_c)$, where θ_c are control policy parameters. For example, if the control policy π were derived from solving a Linear Quadratic Regulator (LQR) problem, θ_c represents a linear state feedback matrix derived from the unique solution to the discrete algebraic Riccati equation. However, 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 $\hat{\mathbf{s}}_\mathbf{F}$.

Control Cost: 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: In order to account for the measurement noise in the inputs of the forecaster and improve generalization to out-of-distribution data, we introduce an adversary in our robotic system which can perturb the inputs of the forecaster. The adversary can be viewed as a hypothetical, virtual agent that perturbs historical training data in order to corrupt forecasts and ultimately make control performance worse. More concretely, the adversary is defined as the map $a : \mathbb{R}^{p\times H} \to \mathbb{R}^{p\times H}$, 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 $\theta_a \in \Theta_a$, where Θ_a is the space of all possible parameters. Additionally, in order to restrict the power of the adversary, we penalize the adversary quadratically for large perturbations: $\|\mathbf{s_H} - \mathbf{s}_\mathbf{H}^{\text{adv}}\|_2^2$.

Overall Cost: The system operates as follows. 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 $\mathbf{s_H}$. Then, the forecaster observes the adversary's perturbed history, $\mathbf{s}_\mathbf{H}^{\text{adv}}$, and predicts the future state of the system, $\hat{\mathbf{s}}_\mathbf{F}$ (1a).

Finally, given the predicted forecast $\hat{\mathbf{s}}_\mathbf{F}$ and the ground-truth forecast $\mathbf{s}_\mathbf{F}$, we calculate the corresponding optimal controls $\hat{\mathbf{u}}$ and $\mathbf{u}$, respectively, for the system using the control policy (1b):

$$\mathbf{s}_\mathbf{H}^{\text{adv}} = a(\mathbf{s}_\mathbf{H}; \theta_a), \quad \hat{\mathbf{s}}_\mathbf{F} = f(\mathbf{s}_\mathbf{H}^{\text{adv}}; \theta_f). \tag{1a}$$

$$\hat{\mathbf{u}} = \pi\left(x_0, \hat{\mathbf{s}}_\mathbf{F}; \theta_c\right), \quad \mathbf{u} = \pi\left(x_0, \mathbf{s}_\mathbf{F}; \theta_c\right). \tag{1b}$$

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

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

For clarity, we introduce the following compact notation for this cost—$\mathcal{J}(\theta_f, \theta_a)$ is the final overall cost $J\left(\mathbf{u}, \hat{\mathbf{u}}, \mathbf{s}_\mathbf{F}, \hat{\mathbf{s}}_\mathbf{F}; x_0\right)$ 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 $\mathbf{s}_\mathbf{F}$. Importantly, the controller's parameters θ_c are determined *implicitly* as the solution to the aforementioned optimal control problem, for each choice of θ_a and θ_f. Therefore, we may focus our attention on optimizing only the forecaster and the adversary parameters. Having defined the information flow in our robotic system, we now formalize the problem addressed in this paper.

Problem 1 (Adversarially-Robust Control)**.** *Given a forecaster f and an adversary a, solve the min-max problem:*

$$\min_{\theta_f} \max_{\theta_a} \quad \mathcal{J}(\theta_f, \theta_a) \tag{3a}$$

$$\textit{subject to:} \quad \mathbf{s}_\mathbf{H}^{adv} = a(\mathbf{s}_\mathbf{H}; \theta_a),\ \hat{\mathbf{s}}_\mathbf{F} = f(\mathbf{s}_\mathbf{H}^{adv}; \theta_f), \tag{3b}$$

$$\hat{\mathbf{u}} = \pi\left(x_0, \hat{\mathbf{s}}_\mathbf{F}; \theta_c\right),\ \mathbf{u} = \pi\left(x_0, \mathbf{s}_\mathbf{F}; \theta_c\right). \tag{3c}$$

Specifically, we aim to find a saddle point of (3) in which the order of the minimum and maximum does not matter.

In Problem 1, the adversary is trying to maximize the overall control cost (3a) by perturbing the original past H measurements, $\mathbf{s}_\mathbf{H}$ (3b). In contrast, the forecaster is trying to minimize the overall cost, (3a), by forecasting the future F measurements, $\hat{\mathbf{s}}_\mathbf{F}$ (3b). Intuitively, this problem captures how to find a forecaster that is robust to adversarial perturbations for reliable robotic decision-making.

IV. Approach

We observe that Problem 1 is a two-player, zero-sum game, and seek both forecaster and adversary parameters which are in equilibrium.

A. Characterizing the Robust Forecasting Game

Here we provide a precise characterization of this game, and introduce key solution concepts.

Player 1 (Forecaster): The forecaster's goal is to predict relevant future system states, despite worst-case perturbations of the history by the adversary. In the lane changing example of Fig. 1, for example, this is the future trajectory of the ego car. Thus equipped, 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 corresponding parameters θ_a which maximize the overall cost in Problem 1, $\mathcal{J}(\cdot, \theta_f)$.

We are now ready to describe relevant solution concepts in Problem 1.

Definition 1 (Global Nash equilibrium)**.** *[16, Defn. 2.1] A pair of actions, θ_f^* and θ_a^*, are a global Nash equilibrium (GNE) of a game if for all θ_f and θ_a in $\Theta_f \times \Theta_a$:*

$$\mathcal{J}(\theta_f^*, \theta_a) \leq \mathcal{J}(\theta_f^*, \theta_a^*) \leq \mathcal{J}(\theta_f, \theta_a^*).$$

Definition 2 (Local Nash equilibrium)**.** *[36, Defn. 1] Let $\|\cdot\|$ denote a vector norm. A pair of actions, θ_f^* and θ_a^*, are a local Nash equilibrium (LNE) of cost function $\mathcal{J}$ if there exists an $\epsilon > 0$ such that for any parameters θ_f and θ_a satisfying $\|\theta_f - \theta_f^*\| \leq \epsilon$ and $\|\theta_a - \theta_a^*\| \leq \epsilon$, we have:*

$$\mathcal{J}(\theta_f^*, \theta_a) \leq \mathcal{J}(\theta_f^*, \theta_a^*) \leq \mathcal{J}(\theta_f, \theta_a^*).$$

A GNE is a point in the space of game strategies where both players cannot change their respective strategy without achieving a less favorable outcome. A LNE is a point in the space of strategies where this property need only hold within a small neighborhood. Definitions 1 and 2 are standard solution concepts in the theory of smooth static games.

LNEs are characterized by the following first- and second-order optimality conditions.

Proposition 1 (First-order Necessary Condition)**.** *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)**.** *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 $\nabla^2_{\theta_a \theta_a} \mathcal{J}(\theta_f, \theta_a) \preceq 0$.*

B. Training the Models

In this work, we use feedforward neural networks (NNs) 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, [30] demonstrates that stochastic gradient descent methods do not necessarily converge to LNE in zero-sum games, but [29] proposes a new second-order gradient update rule that does guarantee convergence to a LNE if one exists. However,

due to 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 [37]. Since we do not use the theoretically-motivated second-order technique of [29], we take care to check the first- and second-order conditions of Propositions 1 and 2 to ensure that we have found a LNE, as described later in Section V-A.

Remark 1 (Robustness to Adversarial Perturbations)**.** *It is readily apparent that $\mathcal{J}(\theta_a^*, \theta_f^*) \leq \mathcal{J}(\theta_a, \theta_f^*)$ when the cost function $\mathcal{J}(\cdot, \theta_f^*)$ is concave in the first argument. However, the examples considered in this work do not exhibit such concavity. Nevertheless, our experimental results demonstrate that, for LNE forecaster parameters θ_f^*, the cost $\mathcal{J}(\cdot, \theta_f^*)$ is substantially robust to adversarial perturbations.*

V. Experiments

We now evaluate our method on two different scenarios. The first is a synthetic Autoregressive Integrated Moving Average (ARIMA) process [38] generated by random parameters. In the second task, we use lane-change data from an autonomous driving scenario with human participants [15]. The experiments aim to demonstrate that 1) our proposed game converges to a LNE and 2) the forecaster trained using our proposed method will be robust to OoD data. We now discuss commonalities between both experiments.

Models: Both the forecaster and adversary are NN models with two fully connected layers and ReLU activations.

Differentiable Model Predictive Control (MPC): In both tasks, 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 linear 1D dynamics for the ARIMA experiment and second-order linear dynamics for the lane-change experiment. For both the experiments, 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. $u_{\min} \leq u_t \leq u_{\max}$. Likewise, we presume that states are also constrained to lie within an axis-aligned box: $x_{\min} \leq x_t \leq x_{\max}$. 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 [39], which allows us to backpropagate derivative information through convex optimization problems and thereby train both the forecaster and adversary end-to-end.

A. Datasets and Benchmark Algorithms

In the experiments, all 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. 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. The task model is then trained on an augmented dataset denoted by $\mathcal{D}_{\text{add}}^{\text{train}} \cup \mathcal{D}_{\text{orig}}^{\text{train}}$.
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.

Experiment Procedure: In both the experiments, 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 (Propositions 1 and 2) to check if the converged parameters θ_f^* and θ_a^* constitute a LNE.

B. ARIMA Forecasting

In order to gain intuition about our method's performance on a small, highly-structured dataset, we first examine time series generated by an ARIMA process. For this experiment we have $n = m = p = 1$, $x_0 = 1$, $A = C = Q = R = 1$, and $B = -1$ and time horizon of $T = 50$. The historical time series is of horizon $H = 25$ and future time series of horizon $F = 25$. Our training dataset is of size $N = 4000$ and the test dataset is of size $N = 1000$. Thus, the training history time series tensor is $\mathbf{s_H} \in \mathbb{R}^{4000 \times 1 \times 25}$. The ARIMA time series is: $s_{t+1} = \mu + \alpha s_t + \beta w_{t-1} + w_t$, where μ is the mean, $w = \mathbb{N}(0, \sigma)$ is noise and α, β are model parameters. μ, α, β are initialized randomly. The white noise variance, σ, is 0.01 for the original dataset and 0.05 for the OoD test dataset. As

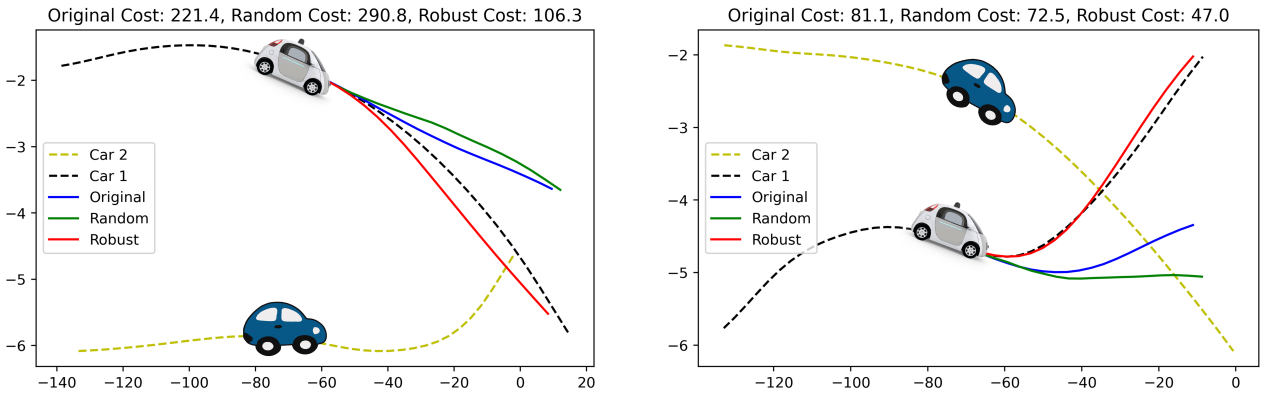

Fig. 2: **Examples of our ROBUST scheme's performance on OoD scenarios:** We show two scenarios from the OoD test dataset. The axes are the scaled Cartesian coordinates of the vehicles. We compare the forecasted lane-change trajectories of ORIGINAL (blue), RANDOM (green), and ROBUST (red) training schemes. The black and yellow dashed lines show the ground-truth trajectories of the ego-vehicle and the other vehicle respectively. All the forecasters were given the historical trajectories of both the vehicles before their current locations. We additionally show the control cost of all forecasted trajectories on the top. The key takeaway is that our ROBUST scheme is able to generate trajectories closer to the ground-truth in OoD scenarios and thus is able to generalize better to unseen scenarios.

such, both the original and the OoD test datasets are generated from quite different distributions. Both hyperparameters λ_f and λ_a were set to 2.0 and were chosen experimentally to balance forecaster performance with control performance while also considering nontrivial adversarial perturbations. Our robust game for the ARIMA forecaster converged to a LNE in approximately 500 iterations.

C. Lane-Change Forecasting

Now, we demonstrate our method's real world applicability on a challenging lane change dataset [40] 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 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 $N = 100$.

In this experiment, the forecaster's goal was to predict the ego vehicle's future trajectory in order to complete a successful lane-change, given the history of states of both cars. The historical time series is of horizon $H = 20$ and 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 time series of both cars. The forecaster's future time series tensor is $\mathbf{s_F} \in \mathbb{R}^{500\times 4\times 20}$. In order 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 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 $\mathcal{D}_{\text{ood}}^{\text{test}}$, 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 are 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. Our robust game for the lane-change forecaster converged to a LNE in approximately 2000 iterations.

D. Results

Our experiments show that our robust forecaster training method reduces the overall cost $\mathcal{J}$ compared to benchmarks.

Qualitative Results: In Fig. 2, we qualitatively demonstrate our method's performance on two OoD lane change scenarios. Specifically, we show trajectories which 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 these OoD scenarios at train time. In these scenarios, two cars are completing a lane-change maneuver. 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. Both the ORIGINAL and RANDOM training schemes are not able to correctly predict the ego-vehicle future trajectory and thus

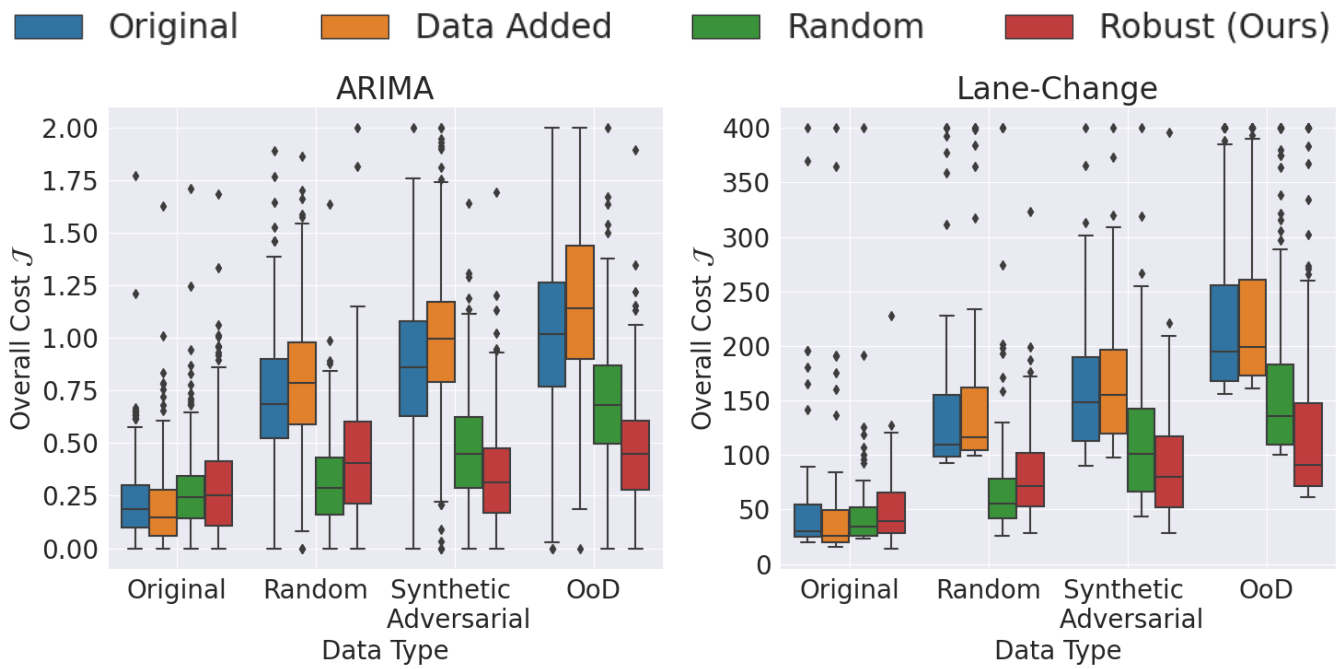

Fig. 3: **Benefits of game-based training for ARIMA and Lane Change Dataset:** We show the overall cost $\mathcal{J}$ for all training schemes on held-out test environments. The x-axis shows different test conditions, and lower overall cost is better. Our ROBUST scheme (red) works on par with other forecasters on the ORIGINAL test data for both experiments. However, it significantly outperforms other forecasters on challenging scenarios in the *out-of-distribution (OoD)* dataset (high-speed lane changes), and *Synthetic Adversarial* test datasets. We beat the RANDOM baseline (green), trained on additional random perturbations, which is not able to model real-world OoD scenarios.

lead to higher control costs. Our ROBUST training scheme is able to correctly predict the ego-vehicle future trajectory and has the lowest control cost.

Quantitative Results: Figure 3 shows the overall cost for all schemes on *test* datasets. On the original test dataset, our ROBUST scheme (red) performs on par with the ORIGINAL (blue) scheme and slightly worse than the DATA ADDED (orange) and RANDOM (green) schemes. The DATA ADDED scheme's performance is expected because it is trained on more original data than our ROBUST scheme, which allows it to perform slightly better on the original test dataset.

However, the key benefits of our approach are shown on the synthetic adversarial test dataset and held-out OoD dataset. 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 which are hard for the forecaster, leading to higher control cost. However, our ROBUST scheme (red) performs significantly better since it was trained to anticipate such unseen perturbations.

In the case of the naturally occurring OoD test dataset, our ROBUST training scheme achieves $\mathbf{50.81}\%$ better performance compared to the RANDOM scheme on the ARIMA dataset. Additionally, on the lane-change dataset, our ROBUST training scheme achieves $\mathbf{30.14}\%$ better performance compared to the RANDOM scheme on the OoD test dataset. The results show that our ROBUST training scheme is able to learn robustness to adversarial scenarios and OoD scenarios. These results are statistically significant using the Wilcoxon signed-rank test [41] with $p < 0.05$ for both datasets. The RANDOM scheme is able to generalize better to OoD data, but is not able to match our ROBUST scheme's performance, since the random perturbations are relatively benign compared to the targeted scenarios generated by our algorithm. The performance of our ROBUST training scheme highlights how our game formulation helps the forecaster generalize better to OoD scenarios. The DATA ADDED scheme has overfit to the original dataset, and consequently performs worse than the ORIGINAL scheme on all other test conditions.

E. Conclusion

In this paper, we considered the challenge of reliable forecasting in robotic decision making. We formulated this problem within the framework of a mathematical game played between a learned forecasting model and a hypothetical adversary which may corrupt prior sensor measurements. Then, we proposed a training scheme which identifies local Nash solutions in this game, and thereby generates more robust forecasts that extend high-quality control performance to unseen OoD scenarios.

Despite the existence of theoretical convergence properties for such problems, we observe that convergence is slow in practice, and sensitive to hyperparameters λ_a, λ_f. Future work should investigate whether structured parameterizations of the forecaster and adversary might permit reliable convergence to a global Nash Equilibrium. Additionally, we will also validate our method on more complicated forecasters, such as LSTM [9] and Transformers [10].

Acknowledgements: This material is based upon work supported in part by the Office of Naval Research (ONR) under Grant No. N000142212254. We also gratefully acknowledge the support of the Lockheed Martin AI Center and Viavi Solutions for this research. Any opinions or findings expressed in this material are those of the author(s). They do not necessarily reflect the views of ONR, the Lockheed Martin AI Center, or Viavi.

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] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] S. Agarwal and S. P. Chinchali, "Synthesizing adversarial visual scenarios for model-based robotic control," in *6th Annual Conference on Robot Learning*, 2022.

[27] 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.

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

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] A. C. Harvey, "Arima models," in *Time Series and Statistics*, J. Eatwell, M. Milgate, and P. Newman, Eds. London: Palgrave Macmillan UK, 1990, pp. 22–24, ISBN: 978-1-349-20865-4.

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

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

[41] W. Conover, *Practical nonparametric statistics*, 3. ed, ser. Wiley series in probability and statistics. New York, NY [u.a.]: Wiley, 1999, VIII, 584, ISBN: 0471160687.