



Delft University of Technology

Document Version

Final published version

Licence

CC BY

Citation (APA)

Schumann, J. F., Hagenus, J., Mathiesen, F. B., & Zgonnikov, A. (2026). Realistic Adversarial Attacks for Robustness Evaluation of Trajectory Prediction Models via Future State Perturbation. *ACM Journal on Autonomous Transportation Systems*, 3(4), Article 23. <https://doi.org/10.1145/3799894>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

In case the licence states "Dutch Copyright Act (Article 25fa)", this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Realistic Adversarial Attacks for Robustness Evaluation of Trajectory Prediction Models via Future State Perturbation

JULIAN SCHUMANN, Department of Cognitive Robotics, TU Delft, Delft, Netherlands

JEROEN HAGENUS, Department of Cognitive Robotics, TU Delft, Delft, Netherlands

FREDERIK MATHIESEN, Delft Center for Systems and Control, TU Delft, Delft, Netherlands

ARKADY ZGONNIKOV, Cognitive Robots, TU Delft, Delft, Netherlands

Trajectory prediction is a key element of autonomous vehicle systems, enabling them to anticipate and react to the movements of other road users. Evaluating the robustness of prediction models against adversarial attacks is essential to ensure their reliability in real-world traffic. However, current approaches tend to focus on perturbing the past positions of surrounding agents, which can generate unrealistic scenarios and overlook critical vulnerabilities. In this work, we demonstrate that perturbing not just past but also future states of adversarial agents can uncover previously undetected weaknesses and thereby provide a more rigorous evaluation of model robustness. Our novel approach incorporates dynamic constraints and preserves tactical behaviors, enabling more effective and realistic adversarial attacks. We introduce new performance measures to assess the realism and impact of these adversarial trajectories. Testing our method on two state-of-the-art prediction models across multiple datasets revealed significant increases in prediction errors and collision rates under adversarial conditions. Qualitative analysis further showed that our attacks can expose critical weaknesses, such as the inability of the model to detect potential collisions in what appear to be safe predictions. These results underscore the need for more comprehensive adversarial testing to better evaluate and improve the reliability of trajectory prediction models for autonomous vehicles.

CCS Concepts: • **General and reference** → *Reliability; Evaluation*; • **Computing methodologies** → *Neural networks*;

Additional Key Words and Phrases: Trajectory prediction, adversarial attack, robustness, automated driving

ACM Reference Format:

Julian Schumann, Jeroen Hagenus, Frederik Mathiesen, and Arkady Zgonnikov. 2026. Realistic Adversarial Attacks for Robustness Evaluation of Trajectory Prediction Models via Future State Perturbation. *ACM J. Auton. Transport. Syst.* 3, 4, Article 23 (July 2026), 26 pages. <https://doi.org/10.1145/3799894>

1 Introduction

Trajectory prediction—the task of forecasting the future positions of traffic participants over time – is a critical component of **autonomous vehicle (AV)** systems, allowing them to adapt to the

Julian Schumann and Jeroen Hagenus, contributed equally to this research.

Authors' Contact Information: Julian Schumann (corresponding author), Department of Cognitive Robotics, TU Delft, Delft, Netherlands; e-mail: J.F.Schumann@tudelft.nl; Jeroen Hagenus, Department of Cognitive Robotics, TU Delft, Delft, Netherlands; e-mail: Jeroen.hagenus@gmail.com; Frederik Mathiesen, Delft Center for Systems and Control, TU Delft, Delft, Netherlands; e-mail: F.B.Mathiesen@tudelft.nl; Arkady Zgonnikov, Cognitive Robots, TU Delft, Delft, Netherlands; e-mail: A.Zgonnikov@tudelft.nl.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2833-0528/2026/07-ART23

<https://doi.org/10.1145/3799894>

actions of nearby road users (Figure 1A). As AVs become more prevalent, ensuring the robustness and reliability of the prediction models is essential for public safety [13, 19]. The use of adversarial trajectories—specifically generated to deceive the model prediction as much as possible—has emerged as a method for evaluating robustness of trajectory prediction models, helping to uncover vulnerabilities that could lead to dangerous situations on the road, particularly in the presence of adversarial agents [4, 5, 15, 24, 30, 36]. The ability of a model to withstand such attacks is known as adversarial robustness [31].

Recent advances in adversarial attack generation have shown promise in exposing system vulnerabilities [4, 10, 30, 33, 36]. Existing attack generation methods have traditionally focused on perturbing the observed position of the target vehicle. These methods can create scenarios that challenge trajectory prediction models, leading to prediction errors or simulated collisions (see Section 2). State-of-the-art approaches demonstrate two key capabilities. First, they generate trajectories that are dynamically feasible, ensuring that attacks can actually be reproduced in a real-world setting [36]. Second, the generated trajectories adhere to traffic norms, ensuring that resulting attacks cannot be trivially detected [4, 36], and ensuring that evaluations of prediction models on the perturbed data are meaningful. However, a key limitation of existing methods is their exclusive focus on perturbing *observed* (past) states of the target vehicle, disregarding the effect of perturbations on the vehicle's future states. This means that existing attack methods can result in adversarial trajectories that restrict the agent to a different *tactical behavior* (maneuver) than the original trajectory, potentially negating the real-life effect of the attack.

As an example, consider an intersection scenario where an adversarial agent intends to turn left, but aims to deceive the ego vehicle (AV across the intersection) into believing that it would *not* turn left (Figure 1B). If the future states of the adversarial agent are not taken into account when generating the attack (as is the case for existing methods), the perturbation of the past trajectory might put the agent in the state from which the left turn is kinematically infeasible. As a result, the adversarial prediction would be consistent with the actual behavior of the agent, meaning that the attack would not actually deceive the ego agent (Figure 1B).

In summary, disregarding future states when generating trajectory perturbations limits the ability of existing attack methods to represent adversarial behaviors that AVs could encounter in the real world. This in turn limits the effectiveness of adversarial testing of trajectory prediction models. The challenge thus lies in generating adversarial trajectories that not only deceive trajectory prediction models, but also remain consistent with the target agent's original tactical behavior throughout the entire trajectory, i.e., for both past and future states.

In this article, we present a novel method for generating adversarial attacks that accounts for both observed and future states. Our approach improves on existing techniques by introducing constraints on future states, ensuring that the entire adversarial trajectory maintains realistic driving behavior (Figure 1C). This allows us to create more subtle and potentially dangerous attack scenarios, such as slight modifications to the path of a vehicle that lead to unexpected collisions while appearing benign to the prediction model (Figure 1D). By considering future states during attack generation, we ensure that adversarial trajectories remain plausible and reflective of real-world driving scenarios, leading to a more comprehensive and realistic evaluation of the robustness of trajectory prediction models. The contributions of this article can be summarized as follows:

- We demonstrate that current methods of generating adversarial trajectories are dynamically infeasible and thus cannot be implemented in real vehicles, and, consequently, have limited value in improving the robustness of trajectory prediction models.
- We introduce a new method to generate dynamically feasible adversarial attacks that are consistent with the observed tactical behavior, allowing for a more meaningful evaluation of the robustness of a prediction model. Building on the approach by Cao et al. [4] to perturb

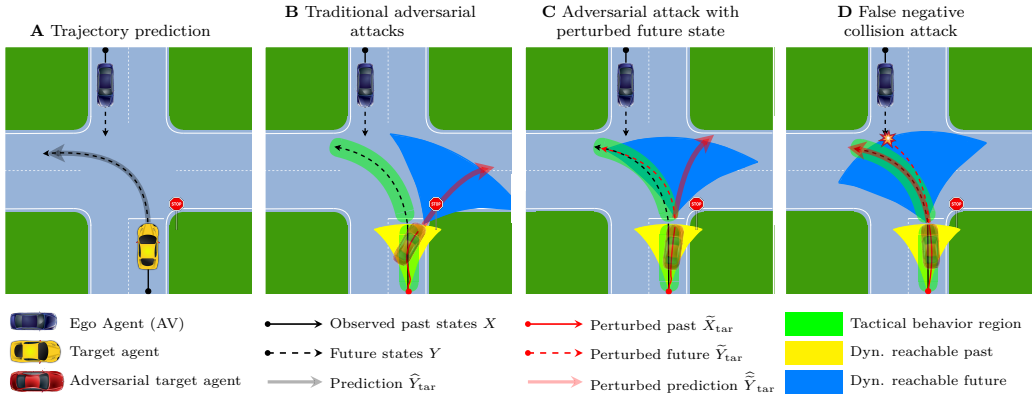


Fig. 1. Scenarios of adversarial attacks by the target agent on trajectory prediction model of the ego agent (AV). (A) In the baseline, non-adversarial scenario, based on the observed past trajectory $X = \{X_{ego}, X_{tar}\}$ of the target agent, the ego agent makes a prediction $\hat{Y}$ that aims to closely capture the (unknown) future states Y . (B) Adversarial (target) agent aims to attack the trajectory prediction model of the AV with the goal for that model to produce an erroneous prediction of the target agent. Existing approaches in the literature typically generate adversarial attacks by perturbing past states of the target vehicle $\tilde{X}$ in such a way that the corresponding prediction $\hat{Y}$ deviates as much as possible from the ground-truth future states Y . However, existing methods do not guarantee that these future states Y actually lie in the reachable set of the target agent after the perturbation, which can result in an overall unsuccessful attack where $\hat{Y}$ is actually consistent with the actual future of the adversarial agent. (C) In our proposed approach, we generate (and constrain) perturbations not only for the past states $\tilde{X}$ but also future states $\tilde{Y}$ of the target agent. This ensures that the observed future Y stays dynamically feasible, which ensures that prediction errors resulting from the perturbation are meaningful. (D) Our approach allows us to generate a novel type of adversarial attacks: *false negative collision attack*, where the adversarial agent takes a future trajectory that collides with the AV but deceives the AV into predicting a non-colliding trajectory. In (B) - (D), the yellow area represents the dynamically reachable set starting from the first observation, while the blue region represents the same starting from the prediction point. The green region represents a simple distance constraint often employed in adversarial attacks in the literature.

not the trajectory in itself, but rather extracted control inputs of a dynamical model, we extend this approach to cases where only position data is available, while considering the agents ability to reverse. Importantly, we increase the meaningfulness of such attacks by constraining the planned perturbed trajectory as well.

- We propose a new type of adversarial attack, the *false negative collision attack*. This goes beyond that standard approach of maximizing the prediction error, and instead aims to cause a collision by fooling the ego agent into discarding such a possibility until it is too late.
- We present novel measures to assess the effectiveness and realism of adversarial attacks on trajectory prediction models, based on the distance between ground truth and perturbed past trajectory as well as the amplitude of the control actions required to execute the perturbed trajectory.

2 Related Work

2.1 Adversarial Attack Generation Methods

Existing methods of adversarial attack generation for trajectory prediction models almost exclusively generate adversarial scenarios by perturbing real traffic scenarios [4, 30, 32, 33, 36] (however, see [22]

for an approach using generative models to sample adversarial scenarios or [20] for backdoor attacks on a models training data). Perturbation-based attack generation methods aim at finding the “worst” possible perturbation for a given scenario and trajectory prediction model. They can be divided into black-box attacks, which only use the output of the model, and white-box attacks, which use gradient information to find perturbations.

Black-box attacks like particle swarm optimization [36], do not require model gradient information but can be slow and unreliable [4]. In contrast, white-box attacks, using some form of gradient descent [4, 30, 33, 36], are generally effective but require the adversarial agent to have access to the model gradient. In cases where access to gradients is restricted, for example due to intellectual property concerns, this challenge can be addressed by using surrogate models and generating transferable attacks [21]. While our work only focuses on utilizing a white-box approach, all those methods could be used to pursue varying attack objectives.

2.2 Adversarial Attack Objectives

Creating adversarial attacks involves specifying the attack objective for robustness evaluation, with different lines of attack being proposed in the literature.

- **Average or final displacement error (ADE/FDE) attack:** Maximize either the average or final distance between the observed ground truth and the prediction made by the behavior model based on the perturbed past [4, 9, 15, 24, 36, 36].
- **Lateral/longitudinal attack.** Maximize the displacement projected either along the trajectory (longitudinal) or perpendicular to it (lateral), to falsely suggest behaviors such as large accelerations or lane changes [8, 9, 15, 36].
- **Targeted attack.** Minimize the distance between the prediction and a user-defined target trajectory [30]. While this method allows for manipulation of the output of the model, the range of possible outcomes is often more constrained compared with other attack methods.
- **Collision attack.** Minimize the distance between the prediction and the ground truth trajectory of the ego vehicle, to deceive the model into predicting a collision [8, 24, 32].

2.3 Constraints Used During Adversarial Attacks

As simple attack objectives like ADE/FDE could be maximized by using very large (and therefore unrealistic and easily detectable) perturbations, previous work generally adopted the approach of limiting the perturbation magnitude to maintain the dynamic feasibility of the attacks. Two approaches to constraining perturbation magnitude have been used in the literature:

- **Constraining vehicle positions:** Zhang et al. [36] use a method referred to as a “Search attack” to constrain trajectories by rescaling an initial perturbation so that velocity, accelerations and jerk lie within fixed bounds based upon the spread of those properties in the original dataset. Yin et al. [33] enhanced this approach by substituting the data statistics method with a continuous curvature model called an “SA-attack” using a pure pursuit method (a path-tracking algorithm that generates a trajectory by continuously pursuing a point on the path at a fixed look-ahead distance from the current position [7]). Namely, a dynamic model is fit between initial perturbations to generate trajectories that ensure dynamic feasibility at specific speeds. As an alternative approach, Cao et al. [4] did not use hard constraints, but instead added a penalty to the attack objective for overly large perturbations.
- **Constraining vehicle control actions:** Cao et al. [4] proposed generating adversarial trajectories by perturbing control actions (with hard constraints) derived from the agent’s spatial position, heading angle, and velocity using a kinematic bicycle model. This ensures both that the trajectories adhere to the physical constraints associated with a specific vehicle

and that the perturbed control actions remain consistent with those of the unperturbed trajectory. Meanwhile, Dong et al. [8] added a regularization term to the attack objective to minimize curvature and its derivative in the perturbed trajectories.

Fusing these two approaches, Fan et al. [10] used the discriminator of a **generative adversarial network (GAN)** to keep the perturbed trajectories similar to the unperturbed one, after which the control inputs of bicycle model were optimized to create a kinematically feasible trajectory matching the generated one as closely as possible.

Our proposed method could generate attacks by perturbing both spatial positions [36] or control actions [4], however, we will focus on the more novel, latter approach in the following sections.

3 Background

3.1 Trajectory Prediction

The goal of trajectory prediction is to forecast the future movements of a traffic agent i based on their past observed trajectory $X_i = \{\mathbf{x}_i(t) \mid t \in \{-H+1, \dots, 0\}\}$, that contains H positions with a time step size Δt . Similarly, we can also collect the future trajectory of the agent $Y_i = \{\mathbf{y}_i(t) \mid t \in \{1, \dots, T\}\}$ with the prediction horizon T . In one scene, $X = \{X_1, \dots, X_N\}$ is the combined history for all N agents in the current scene (similarly, Y is the combined future).

A probabilistic model $\mathbb{P}_\theta$ with trainable parameters θ then generates K equally likely trajectories $\hat{Y}^k \sim \mathbb{P}_\theta(\cdot|X)$ with $\hat{Y} = \{\hat{Y}^k \mid k \in \{1, \dots, K\}\}$ from the same initial state X . Generally, such models are trained to minimize the distance of those trajectories to the ground truth observation Y , or to maximize the likelihood $\mathbb{P}_\theta(Y|X)$ of Y .

While there are models that actually predict the trajectories of all agents simultaneously [12, 23, 34], most models predict agents independently, that is, they have the form $\mathbb{P}(Y_i \mid X_i, X \setminus X_i)$.

3.2 Adversarial Attacks

In the case of an adversarial attack, an adversarial agent, also referred to as a target agent with trajectory X_{tar} , attempts to deceive the prediction model employed by the autonomous vehicle (ego agent X_{ego}) into making incorrect predictions (Figure 1B).

As discussed in Section 2, most existing attack approaches generate adversarial trajectories by modifying an existing non-adversarial trajectory. Specifically, the observed (past) states of the target agent are modified using perturbations δ_X , resulting in the perturbed past trajectory $\tilde{X}_{\text{tar}} = X_{\text{tar}} + \delta_X$. The attacked model then makes the (generally probabilistic) predictions $\hat{Y}_{\text{tar}} \sim \mathbb{P}_\theta(\cdot|\tilde{X}_{\text{tar}}, X_{\text{ego}})$, which are then used to evaluate the robustness of the models.

3.3 Projected Gradient Descent

To generate adversarial attacks using a white-box attack strategy, we use **Projected Gradient Descent (PGD)**. PGD is an extension of traditional gradient descent methods for constrained optimization problems, iteratively refining the perturbations using alternating gradient and projection steps [27]. Given an initial perturbation $\delta^0 = \mathbf{0}$, we perform two steps in each iteration $m \in \{1, 2, \dots, M_{\text{max}}\}$. First, a gradient step based on the loss function $\mathcal{L}$ utilizes a step size α to update the perturbation δ^{m-1} , producing an intermediate perturbation $\hat{\delta}^{m-1}$:

$$\hat{\delta}^{m-1} = \delta^{m-1} - \alpha^{m-1} \nabla_{\delta} \mathcal{L}. \quad (1)$$

Second, a projection step finds an updated perturbation δ^m by projecting $\hat{\delta}^{m-1}$ onto the feasible set C (Equation 2).

$$\delta^m = \underset{v \in C}{\operatorname{argmin}} \|v - \hat{\delta}^{m-1}\| \quad (2)$$

3.4 Adversarial Attack Objectives

In this article, we assume that the loss function $\mathcal{L}$ of the attacked model can be a sum of multiple objective or constraint functions l . The following attack objectives l are common in the literature:

- **ADE/FDE attack** tries to maximize the average or final distance between ground truth Y and adversarial predictions $\hat{\tilde{Y}}$ [4, 9, 15, 36].

$$\begin{aligned} l_{\text{ADE}}(Y_{\text{tar}}, \hat{\tilde{Y}}_{\text{tar}}) &= -\frac{1}{KT} \sum_{k=1}^K \sum_{t=1}^T \left\| \hat{\tilde{\mathbf{y}}}_{\text{tar}}^k(t) - \mathbf{y}_{\text{tar}}(t) \right\| \\ l_{\text{FDE}}(Y_{\text{tar}}, \hat{\tilde{Y}}_{\text{tar}}) &= -\frac{1}{K} \sum_{k=1}^K \left\| \hat{\tilde{\mathbf{y}}}_{\text{tar}}^k(T) - \mathbf{y}_{\text{tar}}(T) \right\| \end{aligned} \quad (3)$$

- **Collision attack** aims to mislead the ego agent into predicting a collision (i.e., a false positive collision prediction) by minimizing the distance between the predicted positions of the target agent $\hat{\tilde{Y}}_{\text{tar}}$ and the ground truth future positions of the ego agent Y_{ego} [24, 32]:

$$l_{\text{Col}}(Y_{\text{ego}}, \hat{\tilde{Y}}_{\text{tar}}) = \frac{1}{K} \sum_{k=1}^K \min_{t \in \{1, \dots, T\}} \left\| \hat{\tilde{\mathbf{y}}}_{\text{tar}}^k(t) - \mathbf{y}_{\text{ego}}(t) \right\|. \quad (4)$$

4 Method

4.1 Perturbation Strategy

To improve on the realism of the traditional approach of perturbing the past states of the target vehicle X_{tar} , in our method we also perturb the future states Y_{tar} with perturbation δ_Y : $\tilde{Y}_{\text{tar}} = Y_{\text{tar}} + \delta_Y$. Because the resulting perturbed trajectories $\tilde{X}_{\text{tar}}$ and $\tilde{Y}_{\text{tar}}$ need to be similar to the original trajectory, perturbation magnitudes are constrained to $\delta_X \in C_X$ and $\delta_Y \in C_Y$. Furthermore, to ensure dynamic feasibility of the perturbed trajectories, we follow Cao et al. [4] and perturb not the recorded positions X_{tar} (which may result in trajectories that cannot be executed by real vehicles), but instead the control inputs $\mathbf{u} = \{a, \kappa\}$ underlying this trajectory. Here, control inputs include longitudinal acceleration a and curvature κ .

4.2 Dynamic Model

To allow us to perturb control actions, we must first be able to extract these control actions from a given trajectory $P = \{\mathbf{p}(t) \mid t \in \{T_0, \dots, T_1\}\}$ (with positions $\mathbf{p}(t) = \{x(t), y(t)\}$), and transform the perturbed control actions into the perturbed trajectory. To this end, we use the dynamic unicycle model Φ , previously utilized by [4]. This dynamic model allows us to advance the state $\mathbf{s}(t) = \{x(t), y(t), \theta(t), v(t)\}$ with heading angle θ and longitudinal velocity v using the control input $\mathbf{u}(t) = \{a(t), \kappa(t)\}$ with acceleration a and curvature κ :

$$\mathbf{s}(t+1) = \Phi(\mathbf{s}(t), \mathbf{u}(t)) . \quad (5)$$

Specifically, we get

$$\begin{aligned} v(t+1) &= v(t) + a(t)\Delta t \\ \theta(t+1) &= \theta(t) + v(t)\kappa(t)\Delta t \\ x(t+1) &= x(t) + v(t+1) \cos(\theta(t+1))\Delta t \\ y(t+1) &= y(t) + v(t+1) \sin(\theta(t+1))\Delta t . \end{aligned} \quad (6)$$

The dynamic model can also be inverted to give us the control actions, with $\mathbf{u}(t) = \Phi^{-1}(\mathbf{s}(t+1), \mathbf{s}(t))$, with

$$\begin{aligned} a(t) &= \frac{v(t+1) - v(t)}{\Delta t} \\ \kappa(t) &= \frac{\theta(t+1) - \theta(t)}{v(t)\Delta t}. \end{aligned} \quad (7)$$

However, many existing vehicle datasets only provide position data [29] but not the velocity or heading data needed for the inverse dynamical model to extract the control inputs (see Equation (7)). While it is possible to extract $\theta(t)$ and $v(t)$ from $\mathbf{p}(t)$ and $\mathbf{p}(t-1)$ based on Equation (6), as we do not know $\mathbf{p}(T_0-1)$, this leaves us unable to determine $\theta(T_0)$ and $v(T_0)$. Consequently, to be able to use the inverse dynamic model at $t = T_0$, we adopt the assumption that, in a given unperturbed trajectory, velocities and heading angles are constant over the first two timesteps. While this assumption is unrealistic, in practice the consequences of its violation are negligible for small timesteps. This allows us then to extract the initial state using Equation (6), with $v_x(t) = \frac{x(t+1)-x(t)}{\Delta t}$ and $v_y(t) = \frac{y(t+1)-y(t)}{\Delta t}$:

$$\begin{aligned} v(T_0) &= v(T_0+1) = \sqrt{v_x(T_0)^2 + v_y(T_0)^2} \\ \theta(T_0) &= \theta(T_0+1) = \text{atan2}(v_y(T_0), v_x(T_0)). \end{aligned} \quad (8)$$

Based on those starting points, we can then extract the later velocities v and heading angles θ based on Equation (6):

$$\begin{aligned} v(t+1) &= \mathcal{D}_t \sqrt{v_x(t)^2 + v_y(t)^2} \\ \theta(t+1) &= \text{atan2}(\mathcal{D}_t v_y(t), \mathcal{D}_t v_x(t)). \end{aligned} \quad (9)$$

Here, $\mathcal{D}_t$ describes the current direction of travel (i.e., if the agent is reversing or driving forward), with

$$\mathcal{D}_t = \begin{cases} \text{sgn}(v(t)) & |\text{atan2}(v_y(t), v_x(t)) - \theta(t)| \leq \frac{\pi}{2} \\ -\text{sgn}(v(t)) & \text{otherwise} \end{cases}. \quad (10)$$

This ensures that reversing the direction of travel is correctly extracted as a (sign) change in velocity brought about by realistic accelerations or decelerations and not as a turn that would require an impossibly high curvature.

Once we have extracted all the values of $v(t)$ and $\theta(t)$ for $t \in \{T_0, \dots, T_1\}$, we can use the inverse dynamical model from Equation (7) to extract the corresponding control inputs $a(t)$ and $\kappa(t)$ for each $t \in \{T_0, \dots, T_1\}$.

4.3 Generating Dynamically Feasible Attacks

With the above dynamic model, we can generate dynamically feasible attacks in an iterative procedure (Figure 2).

- (1) In the first iteration ($m = 0$), we extract the control actions $U_{\text{tar}} = \{\mathbf{u}_{\text{tar}}(t) \mid t \in \{-H+1, \dots, -1\}\}$ and $V_{\text{tar}} = \{\mathbf{u}_{\text{tar}}(t) \mid t \in \{0, \dots, T-1\}\}$ from the unperturbed trajectories X_{tar} and Y_{tar} respectively. After initializing the added states (Equation (8)), we can use alternately the inverse model Φ^{-1} (Equations (7) and (9)) to get the current control actions, before using the forward model Φ (Equation (6)) to get the next values of the added states (Figure 2.1). We also initialize our control action perturbations $\delta_{U_{\text{tar}}}^0 = \mathbf{0}$ and $\delta_{V_{\text{tar}}}^0 = \mathbf{0}$.
- (2) We get the perturbed control states $\tilde{U}_{\text{tar}}^m = U_{\text{tar}} + \delta_{U_{\text{tar}}}^m$ and $\tilde{V}_{\text{tar}}^m = V_{\text{tar}} + \delta_{V_{\text{tar}}}^m$. These can then be used to compute the corresponding perturbed trajectories $\tilde{X}_{\text{tar}}^m$ and $\tilde{Y}_{\text{tar}}^m$, using the full initial

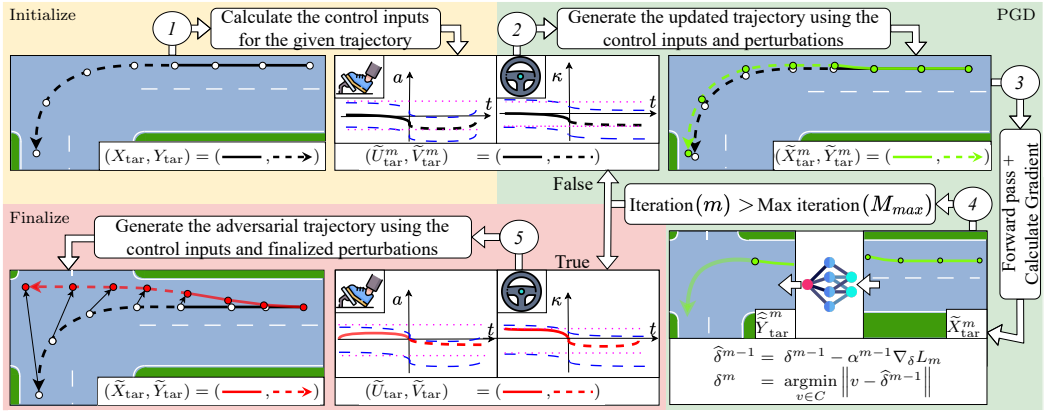


Fig. 2. Generating adversarial trajectories with constraining future states using control actions. (1) During initialization, we extract the control actions ($U_{tar} = \tilde{U}_{tar}^0$, $V_{tar} = \tilde{V}_{tar}^0$) from the unperturbed trajectory parts X_{tar} and Y_{tar} respectively. (2) Afterwards, we iteratively perturb those control actions, where, starting with the given perturbed control actions $\tilde{U}_{tar}^m$ and $\tilde{V}_{tar}^m$, we get the corresponding perturbed trajectories $\tilde{X}_{tar}^m$ and $\tilde{Y}_{tar}^m$. (3) Based thereon – together with the prediction $\tilde{Y}_{tar}^m$ from updated states $\tilde{X}_{tar}^m$ – we can calculate the adversarial loss L_m and its derivative, with which the perturbation can be updated. The constraints C in this update are represented by the purple dotted line for the absolute control limits and the blue dashed line for the relative bounds. (4) This procedure is repeated for M_{max} iterations. (5) In the end, the final perturbed trajectories are extracted.

state $s_{tar}(-H + 1)$ (Equation (8)) and the given control inputs together with the dynamic model Φ (see Figure 2.2).

- (3) We use the prediction model to generate the corresponding predictions $\tilde{Y}_{tar}^m$ based on the perturbed input. Given an adversarial loss function $L^m = \mathcal{L}(X, Y, \tilde{X}_{tar}^m, \tilde{Y}_{tar}^m, \tilde{Y}_{tar}^m)$, we can get the gradient of the loss in regard to the perturbation $\delta_{U_{tar}}^m$ and $\delta_{V_{tar}}^m$ using differentiable programming (see Appendix A for details). With this gradient, we can use projected gradient descent to update the perturbations $\delta_{U_{tar}}^m$ and $\delta_{V_{tar}}^m$ (Figure 2.3).

In regard to the constraints C_U and C_V (Equation (2)), we choose the following values to restrict the perturbations of acceleration $a(t)$ and curvature $\kappa(t)$

$$\begin{aligned}
 \delta_a(t) &\geq \max\{-2 \text{ ms}^{-2}, a_{\min} - a(t)\} \\
 \delta_a(t) &\leq \min\{2 \text{ ms}^{-2}, a_{\max} - a(t)\} \\
 \delta_\kappa(t) &\geq \max\{-0.05 \text{ m}^{-1}, -0.2 \text{ m}^{-1} - \kappa(t)\} \\
 \delta_\kappa(t) &\leq \min\{0.05 \text{ m}^{-1}, 0.2 \text{ m}^{-1} - \kappa(t)\}
 \end{aligned} \tag{11}$$

This approach combines both absolute bounds (i.e., bounds on the resulting perturbed control inputs $\tilde{a}(t)$ and $\tilde{\kappa}(t)$) as well as relative bounds (directly on perturbations $\delta_a(t)$ and $\delta_\kappa(t)$, taken from [4]). For the absolute bounds $a_{\min}$ and $a_{\max}$, we use the lowest and highest acceleration values observed in the dataset.

- (4) If $m > M_{max}$, we complete the attack, having arrived at our final perturbed trajectories $\tilde{X} = \tilde{X}^{M_{max}}$ and $\tilde{Y} = \tilde{Y}^{M_{max}}$ (Figure 2.5).

4.4 Preserving Tactical Behavior

As discussed in Section 1, in order for the adversarial attack to be effective, perturbed trajectories need to provide for the same tactical behavior as the ground truth, i.e., they should be close to the original observations. However, our choice of perturbing control actions instead of positions makes it impossible to implement this via the constraints C_U and C_V (Equation (11)). Therefore, we chose to instead implement such position constraints by adding a regularizing term l to the adversarial loss $\mathcal{L}$. Here, the regularizing term l constrains a whole trajectory by combining costs l_p that constrain each individual perturbed position $\tilde{\mathbf{p}}(t)$ of the original trajectory $P = \{\mathbf{p}(t) \mid t \in \{T_0, \dots, T_1\}\}$.

4.4.1 Regularizing Individual Points in the Perturbed Trajectory. Inspired by negative log-barrier constraint encoding used in interior point methods [2], we use the following regularization term:

$$l_{p, \text{Method}}(\tilde{\mathbf{p}}(t), P) = -\ln(d_{\max} - d_{\text{Method}}(\tilde{\mathbf{p}}(t), P)). \quad (12)$$

Importantly, $l_{p, \text{Method}}(\tilde{\mathbf{p}}(t), P)$ grows to infinity as $d_{\text{Method}}(\tilde{\mathbf{p}}(t), P)$ approaches $d_{\max}$. Therefore, with sufficiently small steps, the constraint $d_{\text{Method}}(\tilde{\mathbf{p}}(t), P) \leq d_{\max}$ will never be violated. Here, the subscript *Method* denotes one of the two options for the choice of the distance function:

– **Time-specific:** The perturbed position is bounded by the corresponding unperturbed position:

$$d_{\text{Time}}(\tilde{\mathbf{p}}(t), P) = \|\tilde{\mathbf{p}}(t) - \mathbf{p}(t)\|. \quad (13)$$

– **Trajectory-specific:** The perturbed trajectory is bounded in its shortest distance to the whole ground truth trajectory; only lateral perturbations are penalized.

$$d_{\text{Traj}}(\tilde{\mathbf{p}}(t), P) = \min_{t'} d_{\text{proj}}(\tilde{\mathbf{p}}(t), \mathbf{p}(t'), \mathbf{p}(t' + 1)). \quad (14)$$

Here, d_{proj} is the closest distance of point onto a line segment spanned by two other points:

$$d_{\text{proj}}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{cases} \|\mathbf{a} - \mathbf{c}\| & r(\mathbf{a}, \mathbf{b}, \mathbf{c}) \leq 0 \\ d_{\perp}(\mathbf{a}, \mathbf{b}, \mathbf{c}) & 0 < r(\mathbf{a}, \mathbf{b}, \mathbf{c}) < 1 \\ \|\mathbf{a} - \mathbf{b}\| & \text{otherwise.} \end{cases}$$

with

$$d_{\perp}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \frac{\|(\mathbf{a} - \mathbf{c}) \times (\mathbf{b} - \mathbf{c})\|}{\|\mathbf{b} - \mathbf{c}\|}$$

$$r(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \frac{(\mathbf{a} - \mathbf{c}) \cdot (\mathbf{b} - \mathbf{c})}{\|\mathbf{b} - \mathbf{c}\|^2}$$

4.4.2 Regularizing Perturbed Trajectories. Based on different combinations of the pointwise constraints discussed above, we propose three different approaches for regularizing full trajectories P (Figure 3):

– **Time-specific:** only use the time-specific constraints (Equation (13)):

$$l_{\text{Time}}(\tilde{P}, P) = \frac{1}{|\tilde{P}|} \sum_{\tilde{\mathbf{p}}(t) \in \tilde{P}} l_{p, \text{Time}}(\tilde{\mathbf{p}}(t), P). \quad (15)$$

– **Trajectory-specific:** only use the trajectory-specific constraints (Equation (14)):

$$l_{\text{Traj}}(\tilde{P}, P) = \frac{1}{|\tilde{P}|} \sum_{\tilde{\mathbf{p}}(t) \in \tilde{P}} l_{p, \text{Traj}}(\tilde{\mathbf{p}}(t), P). \quad (16)$$

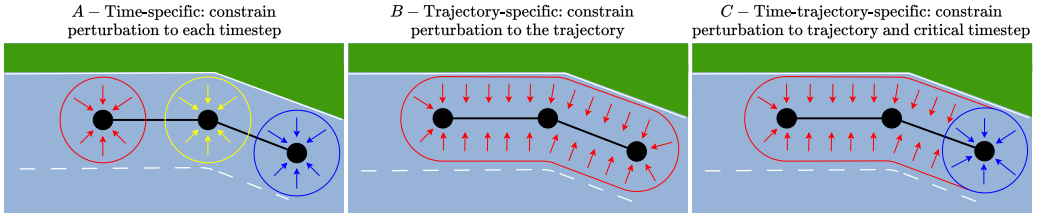


Fig. 3. Three regularization approaches for preserving tactical behavior. The black line represents the ground truth trajectory, with the colored arrows showing the direction in which the regularization penalties are applied. In A), the different colors represent that the different penalties are applied to different timesteps.

- **Time-trajectory-specific:** While the trajectory-specific approach allows for more flexibility in longitudinal deviations, thereby enabling the exploration of a broader range of realistic adversarial scenarios, this also poses a risk. Namely, perturbations may lead to substantial deviations from the ground truth states at critical moments, resulting in unrealistic or unsafe behaviors not aligned with the natural flow of the trajectory. For example, when making a left turn (Figure 1A), it is crucial that, at prediction time $t = 0$, the target agent can still feasibly complete the turn without risking a collision with the ego agent. Therefore we introduce a combined regularization function

$$l_{\text{Time-traj}}(\tilde{P}, P) = l_{\text{Traj}}(\tilde{P}, P) + l_{p, \text{Time}}(\tilde{P}(T_1), P). \quad (17)$$

4.5 Adversarial Attack Objectives

For our new attack generation approach, we implemented the attack objectives common in the literature: ADE/FDE and collision rate (Section 3.4). Furthermore, we also propose a novel attack objective: *false negative collision*. In contrast to the traditional (*false positive*) collision attack (which tries to deceive the AV into predicting a collision), our novel attack tries to conceal an intended collision from the AV. Compared with the false positive collision loss in Equation (4), the attack objective here is

$$l_{\text{FNC}}(Y_{\text{ego}}, \tilde{Y}_{\text{tar}}, \hat{Y}_{\text{tar}}) = \min_{t \in \{1, \dots, T\}} \left\| \tilde{\mathbf{y}}_{\text{tar}}^k(t) - \mathbf{y}_{\text{ego}}(t) \right\| + \frac{1}{T} \sum_{t=1}^T \left\| \frac{1}{K} \sum_{k=1}^K \hat{\mathbf{y}}_{\text{tar}}^k(t) - \tilde{\mathbf{y}}_{\text{tar}}^k(t) \right\|. \quad (18)$$

The first term here is designed to create a perturbed future that seeks a collision, while the second term tries to create a perturbation for which the attacked model's predictions are similar to the prediction resulting from the ground truth (i.e., the prediction model should not be able to differentiate between them). The approach of deceiving the prediction model into making specific predictions was originally proposed by Tan et al. [30]. The value of this novel attack is that it presents a far more dangerous scenario for an AV in actual traffic than the traditional collision attack. Namely, in the false positive collision case, the reaction of the AV would be an unnecessary braking maneuver, while a successful false negative collision attack can be far more severe, especially if the vehicles are close enough that there is not enough time left for a safe evasion upon the AV recognizing the attack.

5 Experimental Setting

We design a number of experiments to test different attack profiles, using STEP—an evaluation framework designed by Schumann *et al.* [29].¹

5.1 Trajectory Prediction Models

For our primary experiments, we applied our attack generation approach to one of the most popular trajectory prediction models: *Trajectron++* [25]. *Trajectron++*, predicting not positions directly, but instead control states of a unicycle model, has the advantage of always producing dynamically feasible predictions, and thus should produce predictions most robust to our dynamically feasible adversarial attacks. We set the model to generate $K = 100$ predictions for each model input, i.e., in each scene.

Additionally, to show that our method is not uniquely suited to just this one model, we also applied it to *ADAPT* [1], a more recent state-of-the-art trajectory prediction model. Compared with *Trajectron++*, it predicts positions directly, further emphasizing the versatility of the attacks.

5.2 Datasets

As a source of unperturbed trajectories for our primary experiments, we chose L-GAP [35], a urban driving dataset obtained in a driving simulator focusing on an intersection scenario. There, sixteen human drivers (yellow vehicle in Figure 1A) executed a combined 1839 valid turns (e.g., no collision), choosing to either go in front of or behind an oncoming AV (blue vehicle), which was programmed to always go straight with constant velocities. We selected the left-turning vehicle as the target agent for which we perturb the trajectory. The oncoming vehicle approaching the intersection was designated as the ego vehicle, which would make a prediction about the behavior of the target agent. Because of noisy measurement of vehicle coordinates in the original dataset, we used the Savitzky-Golay filter [26] to smooth the trajectories.

However, this focus on left turns may introduce a data bias (the actions are either left turn or waiting), a prediction model trained on this dataset alone would have difficulties properly handling different manoeuvres which might arise from adversarial attacks. To address this issue, we trained *Trajectron++* simultaneously on both L-GAP and NuScenes [3], with the latter being larger and including a far more varied range of scenarios. However, adversarial attacks were only generated using trajectories from L-GAP to facilitate interpretation of evaluation results.

In this work, for L-GAP, we used an observation period $H = 12$ time steps, and a prediction horizon $T = 12$ time steps, with a time interval of $\Delta t = 0.1$ s. The prediction time was set so that the computer-controlled ego agent was $T\Delta t$ (i.e., 1.2 s for this dataset) away from the intersection, to ensure that the prediction of the human (adversarial vehicle) would be significant to the ego agent's own path planning (i.e., a wrong prediction would make the ego agent vulnerable to a potentially unavoidable collision). To ensure that the predictions are meaningful, we removed those scenarios where the target agent already initiated the turn (e.g., for L-GAP, we analyzed the remaining 53 scenarios where the agent turned and 971 scenarios where the agent chose to wait).

To test the generalizability of our proposed attack, we also used two additional dataset to first train and then attack and evaluate models on, specifically *highD* [17] (specifically a subset focusing on lane changes [28]) and *roundD* [18]. These datasets cover real-world highways (*highD*) and roundabouts (*roundD*) in Germany. For *highD*, the vehicle in the left lane was considered as the ego vehicle, with the adversarial target agent deciding on whether to merge into the lane in front of ego or not. For *roundD*, the vehicle inside the roundabout was considered as the ego vehicle that

¹The code for running these experiments can be found online: <https://github.com/julianschumann/General-Framework-Adversarial-Attack>

predicts the trajectory of the adversarial target vehicle entering the roundabout (*i.e.*, similar to the previous dataset, the ego agent has priority). Similar to *L-GAP*, both datasets represent gap acceptance problems [28], where the target vehicle can either go first (accept the gap) or yield and go second (reject the gap); see [28] for a discussion on the equivalence of those three scenarios. We extracted the evaluation scenarios from *round* using the same parameters H , T , and Δt as for *L-GAP*, allowing us to evaluate our attack on 174 scenarios where the target agent chose to enter the roundabout first and 2809 scenarios where it chose to wait for the ego agent to pass. Meanwhile, for *highD*, where gaps are normally larger due to the small relative velocities, we used $H = 12$, $T = 25$, and $\Delta t = 0.2s$, resulting in 46 examples where the target agent changed the lane and 543 where it let itself be overtaken.

5.3 Comparison to Established Attack Approaches

We compared our method of perturbing control actions with constraints on future states to two alternative approaches commonly used in the literature [13]. In the first approach, one will simply perturb positions, without any constraints regarding dynamic feasibility. The second approach is the *search attack* proposed by Zhang et al. [36], which also perturbs spatial positions but limits the perturbation using data statistics. Unlike our approach, both of these approaches impose constraints only on past states.

For each of the three attack strategies (positions; positions with data statistics [36]; actions with future state constraints (ours)), we tested all possible combinations of four attack objectives (ADE; FDE; false positive collision; false negative collision) and two constraint types (time-specific; time-trajectory-specific). This resulted in eight adversarial attack loss functions for each attack approach.

Our approach includes perturbing not only observed (past) but also future states; for perturbing the latter, we used the trajectory-specific displacement constraints (Equation (16)). The reason for using only this constraint type is that the future perturbed trajectories $\tilde{Y}_{tar}$ should mostly preserve the actually chosen maneuver, for which this looser constraint is enough.

Finally, to highlight the added value of perturbing control actions with trajectory-specific constraints on future states, we compared our approach to its ablated version attacking control actions without constraining future states.

5.4 Adversarial Attack Parameters

For all loss functions, we set the maximum displacement to $d_{max} = 0.9$ m. This choice is based on a typical lane width of 3.5 m and vehicle width of 1.7 m, so that even after perturbations, a vehicle should not leave its lane. To investigate the sensitivity of our approach to variations of this parameter, we varied it for *highD* with $d_{max} \in \{0.3 \text{ m}, 0.6 \text{ m}, 0.9 \text{ m}\}$.

To generate attacks with projected gradient descent, we used an initial learning rate $\alpha^0 = 0.01$ (Equation (1)), with an exponential decay of $\gamma = 0.99$ (*i.e.*, $\alpha^m = \gamma \alpha^{m-1}$) for $M_{max} = 100$ iterations. Crucially, α can be adjusted if the perturbed observed and future states fall outside the tactical behavior constraint region (*i.e.*, in Equation (12), we see $d(\tilde{p}(t), P) > d_{max}$). In such cases, we repeatedly halved α until this condition was met. This learning rate scaling was applied only within each iteration, so that the following iteration would still use the same learning rate as if no adjustment was necessary on the previous iteration.

5.5 Performance Measures

We employed eight performance measures to evaluate our proposed attack approach as well as established methods. We used measure values *averaged* across all the samples in the dataset.

Attack efficiency. We adopted three measures that are widely used in the literature [13]:

– Average displacement error:

$$\text{ADE} = \frac{1}{KT} \sum_{k=1}^K \sum_{t=1}^T \left\| \widehat{\mathbf{y}}_{\text{tar}}^k(t) - \mathbf{y}_{\text{tar}}(t) \right\|. \quad (19)$$

– Final displacement error:

$$\text{FDE} = \frac{1}{K} \sum_{k=1}^K \left\| \widehat{\mathbf{y}}_{\text{tar}}^k(T) - \mathbf{y}_{\text{tar}}(T) \right\|. \quad (20)$$

– Collision rate:

$$\text{CR}_{\text{pred}} = \frac{1}{K} \sum_{k=1}^K \max_{t \in \{1, \dots, T\}} f_{\text{coll}} \left(\widehat{\mathbf{y}}_{\text{tar}}^k(t), \mathbf{y}_{\text{ego}}(t) \right), \quad (21)$$

with

$$f_{\text{coll}}(\mathbf{a}, \mathbf{b}) = \begin{cases} 1 & \text{BB}(\mathbf{a}) \cap \text{BB}(\mathbf{b}) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}, \quad (22)$$

where $\text{BB}(\cdot)$ represents the bounding box of a vehicle.

To accompany our novel *false negative collision* attack, we propose a new measure

– Rate of collisions along the perturbed path:

$$\text{CR}_{\text{FNC}} = \max_{t \in \{1, \dots, T\}} f_{\text{coll}} \left(\widetilde{\mathbf{y}}_{\text{tar}}(t), \mathbf{y}_{\text{ego}}(t) \right). \quad (23)$$

Perturbation magnitude. We used two measures to measure the extent of the produced perturbations, with larger values corresponding to more noticeable attacks:

– Maximum displacement:

$$D_{\text{max}} = \max_{\{t \in -H+1, \dots, 0\}} \left\| \widetilde{\mathbf{x}}_{\text{tar}}(t) - \mathbf{x}_{\text{tar}}(t) \right\|. \quad (24)$$

– Mean displacement:

$$D_{\text{mean}} = \frac{1}{H} \sum_{t=-H+1}^0 \left\| \widetilde{\mathbf{x}}_{\text{tar}}(t) - \mathbf{x}_{\text{tar}}(t) \right\|. \quad (25)$$

Dynamic feasibility. We used two measures to evaluate the dynamic feasibility of the perturbed past observations $\widetilde{\mathbf{X}}_{\text{tar}}$ using the perturbed control inputs in $\widetilde{\mathbf{U}}_{\text{tar}}$:

– Acceleration magnitude:

$$a = \frac{1}{H-1} \sum_{t=-H+1}^{-1} |\widetilde{a}(t)|. \quad (26)$$

– Curvature magnitude:

$$\kappa = \frac{1}{H-1} \sum_{t=-H+1}^{-1} |\widetilde{\kappa}(t)|. \quad (27)$$

6 Results

We evaluated attack efficiency, the magnitude of generated perturbations, and dynamic feasibility of adversarial trajectories for three attack approaches: position-based attack (Table 1), search attack perturbing positions using data statistics-based constraints (Table 2), and our approach of perturbing control action with constraints on future states (Table 3).

Table 1. Evaluation of *Trajectron++* on *L-GAP* against Perturbed Positions (without using Constraints on Future Behavior)

Attack loss	Pos. constraints (observed)	Attack efficiency				Pert. magnitude		Dyn. feasibility	
		ADE	FDE	CR _{pred}	CR _{FNC}	$D_{\max}$	D_{mean}	a	κ
Unperturbed	–	0.043	0.100	0.635	–	–	–	0.812	0.433
ADE	Time	4.045	3.447	0.007	–	0.770	0.195	53.407	20.860
	Time-traj	3.447	6.203	0.011	–	0.681	0.168	47.066	25.222
FDE	Time	3.983	7.158	0.008	–	0.766	0.186	51.868	23.368
	Time-traj	4.035	7.262	0.008	–	0.846	0.197	56.280	22.338
False Positive	Time	1.371	2.495	0.020	–	0.467	0.119	36.695	27.427
Collision	Time-traj	1.389	2.530	0.020	–	0.495	0.125	39.004	29.138
False Negative	Time	0.562	1.020	0.019	0.736	0.423	0.116	36.427	25.177
Collision	Time-traj	0.598	1.085	0.020	0.741	0.467	0.126	40.322	25.739

The metrics ADE, FDE, $D_{\max}$, and D_{mean} are given in meter, while for the dynamic feasibility, we have $[a] = \text{m s}^{-2}$ and $[\kappa] = \text{m}^{-1}$.

Table 2. Evaluation of *Trajectron++* on *L-GAP* Against the Position-Based “Search” Attack [36] (Without using Constraints on Future Behavior)

Attack loss	Pos. constraints (observed)	Attack efficiency				Pert. magnitude		Dyn. feasibility	
		ADE	FDE	CR _{pred}	CR _{FNC}	$D_{\max}$	D_{mean}	a	κ
Unperturbed	–	0.270	0.548	0.717	–	–	–	0.812	0.433
ADE	Time	3.073	5.521	0.007	–	0.714	0.177	49.392	20.100
	Time-traj	2.028	3.668	0.010	–	0.539	0.131	38.738	27.571
FDE	Time	2.283	4.124	0.007	–	0.594	0.148	43.665	23.848
	Time-traj	2.306	4.165	0.007	–	0.623	0.152	45.145	23.805
False Positive	Time	1.792	3.241	0.014	–	0.514	0.123	36.494	27.938
Collision	Time-traj	1.819	3.290	0.015	–	0.548	0.130	38.991	25.836
False Negative	Time	0.878	1.586	0.020	0.709	0.452	0.122	38.110	24.988
Collision	Time-traj	0.933	1.685	0.019	0.709	0.481	0.130	40.906	24.469

The metrics ADE, FDE, $D_{\max}$, and D_{mean} are given in meter, while for the dynamic feasibility, we have $[a] = \text{m s}^{-2}$ and $[\kappa] = \text{m}^{-1}$.

6.1 Position-Based Attacks Produce Dynamically Impossible Trajectories

We found that both the position attack and the search attack yielded significantly higher ADE and FDE values compared to our approach. For example, ADE attack with time-specific position constraints resulted in the ADE value of 4.045 m for the position-based attack, 3.073 m for the search attack, and only 1.030 m/0.250 m (without/with constraints on future positions) for an attack on control actions, compared to 0.115 m that the model achieves on unperturbed inputs.

However, the high efficiency of position-based attacks was achieved by producing trajectories that are dynamically impossible. For both established methods (Tables 1, 2), the *average* acceleration magnitude a exceeds $30 \text{ ms}^{-2} \approx 3g$ for all attack losses with average trajectory curvature of at least 20 m^{-1} (requiring a turn radius lower than 0.05 m). Trajectories with such extreme acceleration and curvature values cannot be reproduced in real vehicles (see Figure 4 for representative examples of adversarial trajectories) and hence do not provide a valid means to assess adversarial robustness of trajectory prediction models.

Table 3. Evaluation of *Trajectron++* on *L-GAP* Against our Proposed Adversarial Attack – Perturbed Control Actions with Trajectory-Specific Constraints on Future States

Attack loss	Pos. constraints		Attack efficiency				Pert. magnitude		Dyn. feasibility	
	Observed	Future	ADE	FDE	CR _{pred}	CR _{FNC}	$D_{\max}$	D_{mean}	a	κ
Unperturbed	–	–	0.115	0.283	0.015	–	–	–	0.510	0.006
ADE	Time	–	1.030	1.604	0.014	–	0.841	0.525	1.658	0.006
	Time	Trajectory	0.250	0.470	0.021	–	0.150	0.054	0.901	0.005
	Time-traj	–	0.738	1.201	0.017	–	0.525	0.153	1.402	0.005
	Time-traj	Trajectory	0.222	0.435	0.021	–	0.113	0.038	0.878	0.005
FDE	Time	–	1.059	1.651	0.015	–	0.856	0.267	1.792	0.007
	Time	Trajectory	0.581	0.943	0.021	–	0.459	0.159	1.112	0.006
	Time-traj	–	0.841	1.360	0.017	–	0.598	0.161	1.644	0.006
	Time-traj	Trajectory	0.503	0.845	0.021	–	0.349	0.108	1.113	0.005
False Positive	Time	–	0.446	0.715	0.021	–	0.360	0.127	1.065	0.006
Collision	Time	Trajectory	0.295	0.504	0.020	–	0.214	0.077	0.915	0.006
	Time-traj	–	0.332	0.570	0.021	–	0.217	0.067	1.005	0.005
	Time-traj	Trajectory	0.255	0.459	0.020	–	0.152	0.049	0.905	0.006
False Negative	Time	–	0.123	0.222	0.018	0.723	0.074	0.028	0.845	0.006
Collision	Time-traj	–	0.105	0.198	0.018	0.723	0.052	0.021	0.855	0.006

Rows with “–” under “Future” display results for perturbing control actions without constraints on future states. The metrics ADE, FDE, $D_{\max}$, and D_{mean} are given in meter, while for the dynamic feasibility, we have $[a] = \text{m s}^{-2}$ and $[\kappa] = \text{m}^{-1}$.

6.2 Constraining Future States Generates Efficient and Realistic Control Action-based Attacks

The above results demonstrate that the established attacks achieve high efficiency but are by far not realistic. In contrast, our approach of perturbing control actions (instead of positions directly) produces actions that are similar in magnitude to the unperturbed data (Table 3, see an example attack in Figure 4), and therefore can be implemented in a real vehicle. Although the impact of the attacks generated by this approach is much lower than for established approaches (as indicated by lower attack efficiency measures), these attacks still lead to substantially decreased accuracy of the attacked model. For instance, the ADE and FDE values achieved by attacks targeting these measures are still substantially higher than for the predictions based on unperturbed trajectories. Importantly, false positive collision attacks for our approach generate similar (compared to position-only constraints) or higher (compared to the search attack) predicted collision rates.

Further investigating the approach of perturbing control actions, we analyzed the effect of using future position constraints on attack performance (Table 3). We found that imposing constraints on future positions (in addition to constraining observed positions) results in perturbed trajectories which are far more similar to the ground truth. For instance, we see an decrease in acceleration magnitude of around 40% for ADE and FDE attacks, and 10% for a false positive collision attack.

Similar effects were observed for the displacement measures. For past-only constraints, trajectories attacking ADE and FDE consistently approached the maximum allowed displacement of 0.9 m (with values of $D_{\max} = 0.841$ m). For attacks generated by constraining future states however, the displacement values are consistently and substantially lower, resulting in adversarial trajectories that closely align with original, unperturbed ones. This is likely because adding the constraints on future perturbed trajectories impose further limits on the latter timesteps of the observed trajectory (compare Figures 6 and 7 for an illustrative example). The direct consequence of these smaller perturbation magnitudes however is the reduced efficiency, resulting in lower ADE and FDE values across all attack objectives.

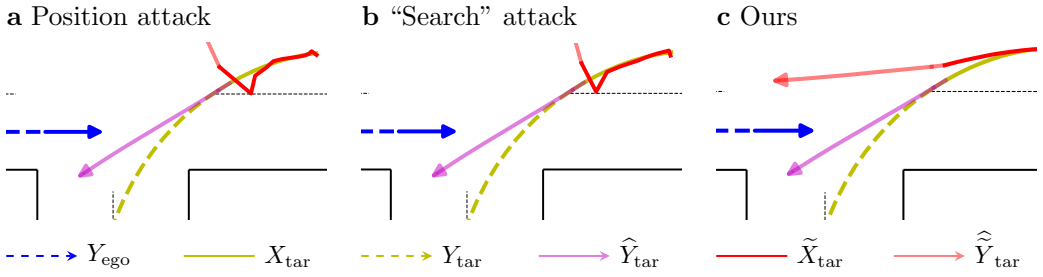


Fig. 4. Examples of adversarial attacks generated using three tested methods on the L-GAP dataset. All three attack methods succeeded in deceiving the ego vehicle's prediction model (*Trajectron++*) into believing that the target vehicle move right or straight, while in fact it was executing a left turn. However, the perturbed trajectories produced by the position attack and the search attack are dynamically infeasible, with near 180 degree reversals. Meanwhile, our method generates a realistic, dynamically feasible trajectory.

These results highlight a trade-off between attack efficiency (how likely an attack is to deceive a prediction model) and obscurity (how difficult it is to detect the attack). However, surprisingly, we found that attacking the collision rate does not conform to this trade-off. Under the false positive collision attack, constraining future states improved the dynamic feasibility and reduced attack magnitude but resulted in similar CR_{pred} values. Even more noteworthy, for ADE and FDE attacks, collision rates actually increased when constraining future states. While surprising, this is likely an artifact of the *L-GAP* dataset. Specifically, in this left turn scenario, the ADE and FDE attacks are optimized by deceiving the model into predicting a turn that is either sharper or wider than the one pursued originally. For the latter option of widening turns, we then see that increased attack success (i.e., even wider turns) can be sufficient to avoid the oncoming vehicle, with the adversarial target agent being predicted to stay in its own lane (see Figure 6 for such an example). However, if future perturbations and therefore prediction errors (as measured by ADE and FDE, see Table 3) are constrained, achieving such sufficiently wide turns becomes less likely, which makes predicted collision more likely (see again Figure 7, where a lower prediction error compared to Figure 6 results in a collision). Consequently, we can expect to see a positive alignment between the prediction errors and the predicted collision rate in datasets with more varied scenarios. An example of this can be found in *round* (see Tables 5 and 6), where in five out of six considered attacks the attacked model with the higher ADE and FDE values also recorded a higher collision rate CR_{pred} .

6.3 Attacking the False Negative Collision Rate is Highly Efficient

Our novel false negative collision attack was remarkably successful in deceiving the model into believing that a perturbed trajectory is not on a collision course, while it actually was. Across all three approaches (Tables 1, 2, 3), at least 70% of generated trajectories were on a collision course while the model believed they were not. Moreover, for the control action-based perturbations (Table 3), the adversarial attacks effectively deceived the model into making predictions $\hat{Y}$ that roughly resemble the predictions based on the ground truth $\hat{Y}$. Here, trajectories generated using the false negative collision loss were indistinguishable from the ground truth, with ADE and FDE values similar or lower than on unperturbed inputs (see also Figure 8). Importantly, highlighting the danger of this attack, the model was deceived only 1.2 s before a potential collision, leaving little if not no time to correct that assessment.

6.4 Regularization Approach Affects Performance of Control Action-based but not Position-based Attacks

For the control action-based attacks (Table 3), using the time-trajectory specific constraints (Equation (17)) generally resulted in lower attack efficiency measures compared to the time-specific approach, while sometimes slightly increasing the collision rates, corresponding to a drop in perturbation magnitude. For example, under the ADE attack with constrained future trajectories (Table 3), the ADE measure drops from 0.250 m to 0.222 m and FDE drops from 0.470 m to 0.435 m, while maximum perturbation decreases from 0.150 m to 0.113 m and average acceleration magnitude from 0.901 ms^{-2} to 0.878 ms^{-2} . On first glance, this might seem somewhat counterintuitive, as the use of the full trajectory-based constraint should allow greater freedom. However, this is not unexpected, as we effectively doubled the constraint cost imposed on the last time step here (Equation (17)), which, due to the constraints on the control actions for dynamic feasibility also constrains the previous time steps, preventing them from exploiting their increased freedom.

This assumption is supported by the fact that we did not observe similar trends in the position-based attacks (Tables 1 and 2), and both constraint types could produce larger or smaller perturbations. Here, the attack can exploit these greater freedoms to produce larger perturbations, as the positions at different time steps are not linked to each other via the dynamic model.

6.5 Attack Performance is Consistent with the Choice of Attack Objective

We analyzed the effect of attack objective on performance measures (Table 3). Intuitively, one would expect that targeting, e.g. ADE would result in largest effect on the ADE measure compared to other attack objectives, yet understanding how other measures (including magnitude and feasibility measures) are affected can provide a better insight into implications of different attacks.

We observed that the FDE loss resulted in larger perturbations than the ADE loss, with the only exception being the mean displacement D_{mean} when not constraining future perturbation and constraining observed ones in a time-specific way. This then also results in higher ADE and FDE values, as well as a higher predicted collision rate CR_{pred} . This is somewhat expected, as the gradient under the FDE loss is driven solely by the time step which generally has the largest displacement, and not averaged over previous time steps with lower displacements.

Meanwhile, the picture for the false positive collision attack is somewhat different. There, attack efficiency in terms of ADE and FDE measures is lower than under the FDE attack, but compared to the ADE attack, these measures are only lower if there is no constraint used for the perturbed future trajectory. If such a constraint is used, the false positive collision attack results in larger ADE and FDE measures compared to the ADE attack. The same relationships can be seen in the perturbation magnitude measures.

6.6 Attack Performance is Consistent Across Models and Datasets and Depends on Perturbation Threshold

To investigate the efficiency of our adversarial attack under different settings, we evaluated a different model (*ADAPT*) on a different dataset (*highD*); furthermore, we varied constraints on the perturbation threshold d_{max} (Equation (12)) in a sensitivity analysis. The results of this analysis (Table 4) are consistent with the above findings: different versions of the attack still managed to deceive the prediction model, with the attacked objectives consistently worsening compared to the unperturbed scenario (e.g., the ADE values under the ADE attack are consistently higher). This confirms that our proposed attack works for different models on different datasets.

Meanwhile, d_{max} has an important influence on the performance of our attack: A higher perturbation budget allows for increasingly more efficient attacks. This is especially noticeable for the ADE attack, where a linear increase in budget leads to increasingly large degradations in model

Table 4. Evaluation of *ADAPT* on *highD* Against our Proposed Adversarial Attack – Perturbing Control Actions with Constraints on Future States – for Different Displacement Thresholds $d_{\max}$ (given in meter)

Attack & constraint (Observed / Future)	$d_{\max}$	Attack efficiency				Pert. magnitude		Dyn. feasibility	
		ADE	FDE	CR_{pred}	CR_{FNC}	$D_{\max}$	D_{mean}	a	κ
Unperturbed	(0.0)	1.612	3.736	0.874	–	–	–	0.576	0.001
ADE	0.3	1.904	4.285	0.902	–	0.189	0.077	0.583	0.001
(Time-traj / Trajectory)	0.6	2.624	5.565	0.907	–	0.393	0.157	0.621	0.001
	0.9	3.509	7.059	0.899	–	0.600	0.236	0.673	0.002
False Positive Collision	0.3	1.807	3.943	0.900	–	0.246	0.118	0.626	0.001
(Time-traj / Trajectory)	0.6	2.180	4.426	0.904	–	0.441	0.204	0.670	0.002
	0.9	2.552	4.911	0.906	–	0.592	0.262	0.712	0.002
False Negative Collision	0.3	1.764	3.978	0.903	0.954	0.178	0.081	0.583	0.001
(Time-traj / –)	0.6	1.999	4.262	0.907	0.941	0.379	0.165	0.596	0.001
	0.9	2.223	4.513	0.909	0.920	0.566	0.249	0.607	0.002

The metrics ADE, FDE, $D_{\max}$, and D_{mean} are given in meter, while for the dynamic feasibility, we have $[a] = \text{m s}^{-2}$ and $[\kappa] = \text{m}^{-1}$.

Table 5. Evaluation of *Trajectron++* on *roundD* with Perturbing Control Actions, with Attacks being Performed Against both *Trajectron++* and *ADAPT*

Attack & constraint (Observed / Future)	Attacked model	Attack efficiency				Pert. magnitude		Dyn. feasibility	
		ADE	FDE	CR_{pred}	CR_{FNC}	$D_{\max}$	D_{mean}	a	κ
Unperturbed		0.043	0.100	0.635	–	–	–	0.812	0.433
ADE	T++	1.058	1.829	0.745	–	0.302	0.115	1.432	0.036
(Time-traj / Trajectory)	<i>ADAPT</i>	1.171	1.921	0.746	–	0.351	0.098	1.256	0.038
False Positive Collision	T++	1.059	1.837	0.744	–	0.229	0.095	1.462	0.036
(Time-traj / Trajectory)	<i>ADAPT</i>	1.023	1.702	0.746	–	0.607	0.175	1.252	0.041
False Negative Collision	T++	0.952	1.732	0.743	0.786	0.203	0.074	1.260	0.033
(Time-traj / –)	<i>ADAPT</i>	1.037	1.828	0.747	0.789	0.084	0.040	1.053	0.029

If the attacked model is the one that is evaluated, it is set in bold. The metrics ADE, FDE, $D_{\max}$, and D_{mean} are given in meter, while for the dynamic feasibility, we have $[a] = \text{m s}^{-2}$ and $[\kappa] = \text{m}^{-1}$.

performance ($d_{\max} : 0.0 \text{ m} \rightarrow 0.3 \text{ m}$ increases ADE by 0.292 m, while $d_{\max} : 0.3 \text{ m} \rightarrow 0.6 \text{ m}$ increases it by 0.720 m, with the increase of 0.885 m for $d_{\max} : 0.6 \text{ m} \rightarrow 0.9 \text{ m}$). In contrast, for the false positive collision attack, increasing the perturbation thresholds leads to increasingly marginal changes in CR_{pred} . However, this is likely caused by the fact that the collision rate is already close to its maximum in the unperturbed case.

Finally, a close linear relationship can be seen between the perturbation threshold $d_{\max}$ and average maximum perturbation $D_{\max}$, with each of the attack producing high R^2 values: $R^2_{\text{ADE}} = 0.9998$, $R^2_{\text{FPC}} = 0.994$, and $R^2_{\text{FNC}} = 0.9998$. This is an indication that our attack attempts to use the entire available displacement budget.

6.7 The Attack Generalizes Across Different Models Trained on the Same Data

To further understand the generalization ability of our adversarial attack, we evaluated it against T++ and ADAPT, both trained on the *roundD* dataset. Importantly, we performed cross-model attacks – that is, attacks that were generated for T++ were tested against ADAPT and vice versa (Tables 5 and 6).

Extending the *L-GAP* and *highD* findings to *roundD*, we found that when attacking the evaluated model directly, the attacks are effective in drastically reducing the model performance, for both *Trajectron++* and *ADAPT*. Similar to *L-GAP*, the false positive collision attack results in worse

Table 6. Evaluation of *ADAPT* on *roundD* with Perturbing Control Actions, with Attacks Being Performed Against Both *Trajectron++* and *ADAPT*

Attack & constraint (Observed / Future)	Attacked model	Attack efficiency				Pert. magnitude		Dyn. feasibility	
		ADE	FDE	CR _{pred}	CR _{FNC}	$D_{\max}$	D_{mean}	a	κ
Unperturbed		0.270	0.548	0.717	–	–	–	0.812	0.433
ADE	<i>T++</i>	0.694	1.062	0.791	–	0.302	0.115	1.432	0.036
(Time-traj / Trajectory)	<i>ADAPT</i>	1.177	1.972	0.803	–	0.351	0.098	1.256	0.038
False Positive Collision	<i>T++</i>	0.555	0.875	0.805	–	0.229	0.095	1.462	0.036
(Time-traj / Trajectory)	<i>ADAPT</i>	1.706	2.636	0.816	–	0.607	0.175	1.252	0.041
False Negative Collision	<i>T++</i>	0.54	0.867	0.814	0.786	0.203	0.074	1.260	0.033
(Time-traj / –)	<i>ADAPT</i>	0.335	0.645	0.812	0.789	0.812	0.040	1.053	0.029

If the attacked model is the one that is evaluated, it is set in bold. The metrics ADE, FDE, $D_{\max}$, and D_{mean} are given in meter, while for the dynamic feasibility, we have $[a] = \text{m s}^{-2}$ and $[\kappa] = \text{m}^{-1}$.

ADE and FDE values than the ADE attack. This might result from the collision attack pushing the predicted trajectory in one specific direction (i.e., towards the ego vehicle), while the ADE attack merely pushes predictions away from the ground truth future. If the unperturbed prediction was well-aligned, with prediction samples falling to either side of the ground truth, the ADE attack may try to push those samples into different directions, thereby minimizing the magnitude of the attack. This effect is further highlighted in Appendix B, which shows a strong correlation between the displacement D and measures such as ADE. A notable difference is, however, that the false negative collision, while still able to produce future perturbations that result in collisions, is much less successful in disguising these intentions, with the predicted collision rate being similar. However, this is expected given the design of the attack, as the predictions are designed to match the unperturbed predictions, which already exhibit a very high collision rate.

Most importantly, cross-model attacks were found to be efficient. Especially *Trajectron++* is still very sensitive against attacks targeted against *ADAPT*, with virtually every evaluation metric being in a similar range or even worse than those resulting from attacks generated for *Trajectron++* itself. This effect is less pronounced on *ADAPT*, where the attacks generated for *Trajectron++* resulted in metric values which, while still worse than for the unperturbed data, are still relatively benign.

6.8 Computational Performance

In our tests, run on a *Intel(R) Core(TM) i9-13980HX* and *Nvidia GeForce RTX 4070*, we saw the following computational performance. When attacking both *Trajectron++* and *ADAPT*, the proposed method generally required between 5 s and 15 s to generate a single attack, depending on the attacked dataset. Consequently, it must be said that in most cases, it is unlikely that such attacks can be generated in real time with the current implementation of our attack.

Nevertheless, it is important to notice that in our implementation, real-time capability was not a goal we pursued. Specifically, it is important to emphasize that the majority of the computational cost arises from backpropagation through the large models involved. As a result, any gradient-based attack is likely to face similar limitations with respect to real-time applicability. One contributing factor to the high runtime is the large number of iterations, $M_{\max} = 100$, used during attack generation. This may be mitigated by introducing convergence criteria to avoid unnecessary iterations. In addition, optimization techniques such as momentum [16] may help achieve faster convergence. Another possible approach is to predict attacks using a neural network trained on a large dataset of paired unperturbed and perturbed trajectories (or the corresponding control inputs). Such predictions could then be refined using a small number of gradient descent iterations to fine-tune the attack. Beyond algorithmic improvements, computational efficiency

could also be increased by reducing implementation overhead through kernel fusion and pre-compilation, improved vectorization and batching, and minimizing host–device memory transfers and synchronizations.

7 Discussion

Our results demonstrate that perturbing control actions with constraints on future positions enables the generation of realistic, feasible, yet still efficient adversarial trajectories. More generally, when testing trajectory prediction models for adversarial robustness, evaluating them just based on attack efficiency measures such as ADE, FDE, or collision rate is insufficient: attack magnitude and dynamic feasibility need to be considered.

Virtually all existing work on adversarial attacks constrains maximum displacements of the perturbed observed trajectories. However, based on our investigation of dynamic feasibility measures, the typically reported resulting drop in model performance is likely not caused by a failure in the prediction model. Instead, such given perturbed inputs lie far outside not only the training domain, but also the realm of physical feasibility, especially when positions – rather than control actions – are perturbed. Consequently, such attacks (which, in addition, could easily be detected in the real world by simple filters on control states) do not provide a viable way of evaluating the robustness of prediction models, unless some effective bounds on the control states are added. Therefore, they do not represent a point of reference to compare the effectiveness of our method against.

Along this line, we have also demonstrated the importance of constraining the perturbations along the future trajectory in addition to the past observation. This has a significant influence on the generated trajectories: while resulting in perturbations of the observed trajectory more comparable to the ground truth, it ensures the consistency between the adversarial agent’s past and future behavior. Without it, comparing the ground truth future Y with the perturbed predictions $\tilde{Y}$ would not actually give any meaningful information on the robustness of the prediction model, as the ground truth Y no longer has any connection to the perturbed past $\tilde{X}$.

One of the key insights of our work is that keeping the adversarial trajectories realistic and dynamically feasible can dramatically reduce attack efficiency in terms of distance-based measures. While our approach is an important step towards finding efficient attack strategies for evaluating adversarial robustness of AVs’ trajectory prediction models, future research needs to focus on improving efficiency of dynamically feasible, realistic, and meaningful attacks.

Complementing our contribution of the new attack approach, our novel attack objective, the *false negative attack*, demonstrated that even with a state-of-the-art prediction model, AVs could still be deceived into a false “sense of safety”, potentially allowing collision to be caused by adversarial agents. Therefore, more research is required to improve robustness of prediction models towards this kind of attacks.

Meanwhile, the standard, (i.e., *false positive*) collision attack did not lead to significant increases in predicted collision rates compared to the ADE and FDE attacks, especially when we constrained the perturbations of future states. While this is somewhat explained by the specific scenario chosen, comparisons to the unperturbed predictions show that further research into this attack is likely needed to improve its potential without compromising its relation to the ground truth data.

Importantly, significant decreases in model performance were shown to be achieved not by attacking the model itself or a surrogate model trained from queries of the original model [14], but instead by simply attacking a model trained on similar data. This means that keeping a (trained) model proprietary may not be a sufficient defense against such attacks, and more robust methods need to be pursued instead.

7.1 Limitations and Future Work

In our experiment, the search attack [36] did not improve dynamic feasibility of the basic position-based attacks, still requiring extreme control actions inconsistent with dynamical model of an actual vehicle. We believe this was mainly caused by the fact that it only uses absolute bounds on perturbed control actions, and does not impose relative bounds. As absolute bounds are dataset-dependent, the use of the *L-GAP* dataset limits our conclusions here. Namely, it includes several collisions which are characterized by high accelerations and curvatures that fall outside normal driving patterns. Consequently, both for the search attack and our approach, it would likely be an improvement to use a more dynamical bound, which could be set to represent some quantile value instead, only increasing for samples where this new bound is surpassed.

Besides this limitation, there are also a number of other improvements and directions that we deem valuable and deserving of their own detailed study in future work. First, given the observed close correlation between the perturbation magnitude and the performance measures ADE and FDE, it is worthwhile to study more closely the influence of those attack hyperparameters that could influence the perturbation magnitude. In particular, future work should evaluate effects of the maximum allowed displacement $d_{\max}$ (Equation (12)), the bounds on control action perturbations δ_a and δ_k (Equation (11)), the length of past observations H , and the number of predicted timesteps. These hyperparameters could also have interdependencies and can affect different models differently. Future research thus needs to systematically investigate their individual and joint effects using a variety of metrics (especially relevant here is predicted collision rate).

Second, given our refined attack, it would be beneficial to study the effectiveness of various defensive responses to this approach. One idea would be to fine-tune prediction models on the perturbed input data. Given that our approach produces inputs that at least kinematically should match the unperturbed data, such a step should be reasonable. However, as this introduces more variance in the data (more inputs map to similar outputs), this will likely degrade the model performance on unperturbed inputs. Similarly, randomized smoothing [6] could be applied to the field of trajectory prediction. As our attack produces relatively small perturbations, such an approach may be feasible, but like the previous idea, risks a degradation of performance on the unperturbed data. Additionally, it may be interesting to test our attack against methods proposed for the detection of adversarial agents [11], as our constraints should produce perturbed trajectories that are less easily discriminated against unperturbed ones.

Third, up to our knowledge, all current research on adversarial attacks on trajectory prediction models is performed in simulation (including the present article). To truly understand the implications of such attacks on automated vehicles, evaluating them in real vehicles would be highly beneficial. However, compared to test in simulation, there are multiple challenges associated with this. Foremost is the need for access to a real-world AV, including not only a prediction model but also the planning algorithm controlling the vehicle. Furthermore, there is a lack of a standardized methodology for implementing and evaluating attacks on real vehicles, as well as safety risks related to real-world testing. Consequently, such an evaluation is out-of-scope for the present article. Still, we believe such research to be necessary in the future to improve robustness of real-world AVs.

8 Conclusion

In this article, we introduced a novel strategy for generating dynamically feasible, realistic, and meaningful adversarial attacks on trajectory prediction models. Our approach successfully deceives the attacked state-of-the-art prediction model, resulting in decreased model accuracy while maintaining plausible behavior of the adversarial agent. Our results highlight the propensity of traditional attacks to produce trajectories which would not be executable in the real world or would not allow the adversarial agent to proceed with its initially intended maneuver. Consequently, we

argue for the need of a comprehensive discussion in the field about the requirements for adversarial attack needed to make them an integral part of the evaluation of the robustness of prediction model. Our work is only the first step in this direction, and we suggest areas of future research that would be beneficial to the evaluation of robustness of autonomous vehicles.

References

- [1] Görkay Aydemir, Adil Kaan Akan, and Fatma Güney. 2023. ADAPT: Efficient multi-agent trajectory prediction with adaptation. 8295–8305. https://openaccess.thecvf.com/content/ICCV2023/html/Aydemir_ADAPT_Efficient_Multi-Agent_Trajectory_Prediction_with_Adaptation_ICCV_2023_paper.html
- [2] Stephen Boyd and Lieven Vandenbergh. 2004. *Convex Optimization*. Cambridge University Press, Cambridge, UK.
- [3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuScenes: A multimodal dataset for autonomous driving. *IEEE Computer Society*, 11618–11628. DOI: [10.1109/CVPR42600.2020.01164](https://doi.org/10.1109/CVPR42600.2020.01164)
- [4] Yulong Cao, Chaowei Xiao, Anima Anandkumar, Danfei Xu, and Marco Pavone. 2022. AdvDO: Realistic adversarial attacks for trajectory prediction. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 36–52. DOI: [10.1007/978-3-031-20065-6_3](https://doi.org/10.1007/978-3-031-20065-6_3)
- [5] Yulong Cao, Danfei Xu, Xinshuo Weng, Zhuoqing Mao, Anima Anandkumar, Chaowei Xiao, and Marco Pavone. 2023. Robust trajectory prediction against adversarial attacks. In *Proceedings of The 6th Conference on Robot Learning*. PMLR, 128–137. <https://proceedings.mlr.press/v205/cao23a.html> ISSN: 2640-3498.
- [6] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*. PMLR, 1310–1320.
- [7] R. Craig Coulter. 1992. *Implementation of the Pure Pursuit Path Tracking Algorithm*. Technical Report CMU-RI-TR-92-01. Carnegie-Mellon University Pittsburgh PA Robotics Institute.
- [8] Yingkai Dong, Li Wang, Zheng Li, Hao Li, Peng Tang, Chengyu Hu, and Shanqing Guo. 2025. Safe driving adversarial trajectory can mislead: Toward more stealthy adversarial attack against autonomous driving prediction module. *ACM Transactions on Privacy and Security* 28, 2 (2025), 1–28.
- [9] Ang Duan, Ruyan Wang, Yaping Cui, Peng He, and Luo Chen. 2023. Causal robust trajectory prediction against adversarial attacks for autonomous vehicles. *IEEE Internet of Things Journal* (2023), 1–1. DOI: [10.1109/JIOT.2023.3342788](https://doi.org/10.1109/JIOT.2023.3342788)
- [10] Jiping Fan, Zhenpo Wang, and Guoqiang Li. 2024. Adversarial attack on trajectory prediction for autonomous vehicles with generative adversarial networks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1026–1031.
- [11] Jiping Fan, Zhenpo Wang, and Guoqiang Li. 2024. A novel unsupervised anomaly detection method on adversarial attacks for autonomous vehicles trajectory prediction. In *2024 IEEE 22nd International Conference on Industrial Informatics (INDIN)*. IEEE, 1–18.
- [12] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D’Souza, Samira E. Kahou, Felix Heide, and Christopher Pal. 2022. Latent variable sequential set transformers for joint multi-agent motion prediction. In *10th International Conference on Learning Representations, ICLR 2022*.
- [13] Jeroen Hagenus, Frederik Baymler Mathiesen, Julian F. Schumann, and Arkady Zgonnikov. 2024. Robustness in trajectory prediction for autonomous vehicles: A survey. In *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 969–976.
- [14] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*. PMLR, 2137–2146.
- [15] Ruochen Jiao, Xiangguo Liu, Takami Sato, Qi Alfred Chen, and Qi Zhu. 2023. Semi-supervised semantics-guided adversarial training for robust trajectory prediction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 8173–8183. DOI: [10.1109/ICCV51070.2023.00754](https://doi.org/10.1109/ICCV51070.2023.00754) ISSN: 2380-7504.
- [16] Diederik P. Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [17] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. 2018. The highD Dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2118–2125. DOI: [10.1109/ITSC.2018.8569552](https://doi.org/10.1109/ITSC.2018.8569552)
- [18] Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. 2020. The rounD dataset: A drone dataset of road user trajectories at roundabouts in Germany. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 1–6. DOI: [10.1109/ITSC45102.2020.9294728](https://doi.org/10.1109/ITSC45102.2020.9294728)
- [19] David Fernández Llorca, Ronan Hamon, Henrik Janklewitz, Kathrin Grosse, Lars Kunze, Patrick Seiniger, Robert Swaim, Nick Reed, Alexandre Alahi, Emilia Gómez, et al. 2024. Testing autonomous vehicles and AI: Perspectives and challenges from cybersecurity, transparency, robustness and fairness. *arXiv preprint arXiv:2403.14641* (2024).

- [20] Mozghan Pourkeshavarz, Mohammad Sabokrou, and Amir Rasouli. 2024. Adversarial backdoor attack by naturalistic data poisoning on trajectory prediction in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14885–14894.
- [21] Yunxiao Qin, Yuanhao Xiong, Jinfeng Yi, and Cho-Jui Hsieh. 2023. Training meta-surrogate model for transferable adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*. DOI : [10.1609/aaai.v37i8.26139](https://doi.org/10.1609/aaai.v37i8.26139)
- [22] Davis Rempe, Jonah Philion, Leonidas J. Guibas, Sanja Fidler, and Or Litany. 2022. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 17305–17315.
- [23] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. 2023. Fjmp: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13745–13755.
- [24] Saeed Saadatnejad, Mohammadhossein Bahari, Pedram Khorsandi, Mohammad Saneian, Seyed-Mohsen Moosavi-Dezfooli, and Alexandre Alahi. 2022. Are socially-aware trajectory prediction models really socially-aware?. In *Transportation Research Part C: Emerging Technologies*, Vol. 141. 103705. DOI : [10.1016/j.trc.2022.103705](https://doi.org/10.1016/j.trc.2022.103705)
- [25] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. 2020. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 683–700. DOI : [10.1007/978-3-030-58523-5_40](https://doi.org/10.1007/978-3-030-58523-5_40)
- [26] Abraham Savitzky and Marcel J. E. Golay. 1964. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry* 36, 8 (1964), 1627–1639.
- [27] Mark Schmidt. 2020. Projected Newton First-Order Optimization Algorithms for Machine Learning Projected-Gradient Methods. <https://www.cs.ubc.ca/~schmidtm/Courses/5XX-S20/S5.pdf>. University of British Columbia, Summer 2020.
- [28] Julian F. Schumann, Jens Kober, and Arkady Zgonnikov. 2023. Benchmarking behavior prediction models in gap acceptance scenarios. *IEEE Transactions on Intelligent Vehicles* 8, 3 (March 2023), 2580–2591. DOI : [10.1109/TIV.2023.3244280](https://doi.org/10.1109/TIV.2023.3244280)
- [29] Julian F Schumann, Anna Mészáros, Jens Kober, and Arkady Zgonnikov. 2025. STEP: Structured training and evaluation platform for benchmarking trajectory prediction models. *arXiv preprint arXiv:2509.14801* (2025).
- [30] Kaiyuan Tan, Jun Wang, and Yiannis Kantaros. 2022. Targeted Adversarial Attacks against Neural Network Trajectory Predictors. DOI : [10.48550/arXiv.2212.04138](https://doi.org/10.48550/arXiv.2212.04138) arXiv:2212.04138 [cs].
- [31] Andrea Tocchetti, Lorenzo Corti, Agathe Balayn, Mireia Yurrita, Philip Lippmann, Marco Brambilla, and Jie Yang. 2025. Ai robustness: A human-centered perspective on technological challenges and opportunities. *Comput. Surveys* 57, 6 (2025), 1–38.
- [32] Jingkan Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. 2021. AdvSim: Generating safety-critical scenarios for self-driving vehicles. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Nashville, TN, USA, 9904–9913. DOI : [10.1109/CVPR46437.2021.00978](https://doi.org/10.1109/CVPR46437.2021.00978)
- [33] Huilin Yin, Jiaxiang Li, Pengju Zhen, and Jun Yan. 2024. SA-attack: Speed-adaptive stealthy adversarial attack on trajectory prediction. In *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1772–1778.
- [34] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. 2021. AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Montreal, QC, Canada, 9793–9803. DOI : [10.1109/ICCV48922.2021.00967](https://doi.org/10.1109/ICCV48922.2021.00967)
- [35] Arkady Zgonnikov, David Abbink, and Gustav Markkula. 2024. Should i stay or should i go? cognitive modeling of left-turn gap acceptance decisions in human drivers. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 66, 5 (May 2024), 1399–1413. DOI : [10.1177/00187208221144561](https://doi.org/10.1177/00187208221144561)
- [36] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z. Morley Mao. 2022. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15159–15168.

Appendices

A Differentiable Programming

We are given the following parts:

- The original trajectories X and Y .
- The control state perturbations δU_{tar} and δV_{tar} , with the resulting perturbed control states $\tilde{U}_{\text{tar}} = U_{X_{\text{tar}}} + \delta U_{\text{tar}}$ and $\tilde{V}_{\text{tar}} = U_{Y_{\text{tar}}} + \delta V_{\text{tar}}$
- The dynamical model Φ , which allows us to respectively combine X and $\tilde{U}_{\text{tar}}$ as well as Y and $\tilde{V}_{\text{tar}}$ to get the perturbed trajectories $\tilde{X}_{\text{tar}}$ and $\tilde{Y}_{\text{tar}}$.

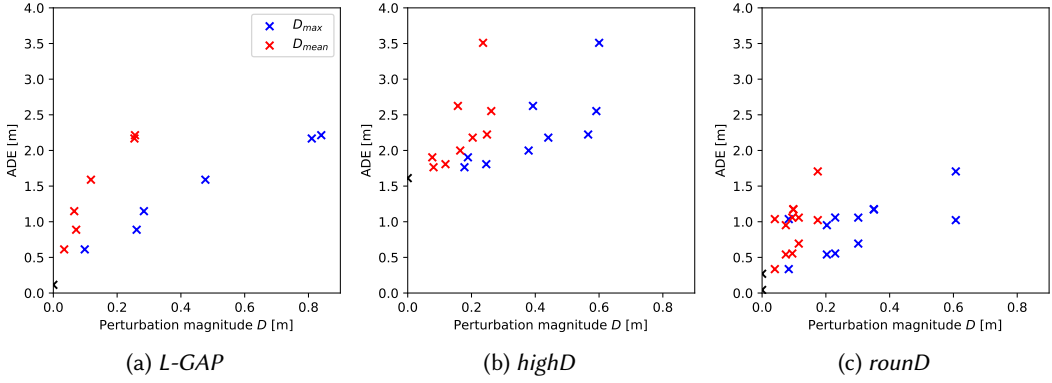


Fig. 5. The relationship between the allowed perturbation magnitude and the resulting ADE values on multiple datasets.

- The prediction model $\mathbb{P}_\theta$, with $\hat{\tilde{Y}} \sim \mathbb{P}_\theta(\cdot | \tilde{X}_{\text{tar}})$
- The adversarial loss function $L = \mathcal{L}(X, Y, \tilde{X}_{\text{tar}}, \tilde{Y}_{\text{tar}}, \hat{\tilde{Y}}_{\text{tar}})$

We then have to find the gradients of the loss function L in regard to the initial control state perturbations. For $\delta_{U_{\text{tar}}}$, we can determine that

$$\begin{aligned}
 \frac{dL}{d\delta_{U_{\text{tar}}}} &= \frac{dL}{d\tilde{U}_{\text{tar}}} \underbrace{\frac{d\tilde{U}_{\text{tar}}}{d\delta_{U_{\text{tar}}}}}_{=1} \\
 &= \frac{dL}{d\tilde{X}_{\text{tar}}} \frac{d\tilde{X}_{\text{tar}}}{d\tilde{U}_{\text{tar}}} \\
 &= \left(\frac{\partial L}{\partial \tilde{X}_{\text{tar}}} + \frac{\partial L}{\partial \tilde{Y}_{\text{tar}}} \frac{d\tilde{Y}_{\text{tar}}}{d\tilde{X}_{\text{tar}}} + \frac{\partial L}{\partial \hat{\tilde{Y}}_{\text{tar}}} \frac{d\hat{\tilde{Y}}_{\text{tar}}}{d\tilde{X}_{\text{tar}}} \right) \frac{d\tilde{X}_{\text{tar}}}{d\tilde{U}_{\text{tar}}}.
 \end{aligned} \tag{28}$$

Here, the calculation of the derivatives $\frac{\partial L}{\partial \cdot}$ depends on the definition of $\mathcal{L}$, the calculation of $\frac{d\tilde{Y}_{\text{tar}}}{d\tilde{X}_{\text{tar}}}$ and $\frac{d\tilde{X}_{\text{tar}}}{d\tilde{U}_{\text{tar}}}$ on the dynamical model Φ , and the calculation of $\frac{d\hat{\tilde{Y}}_{\text{tar}}}{d\tilde{X}_{\text{tar}}}$ on the prediction model $\mathbb{P}$. Similarly, we can do the same for $d\delta_{V_{\text{tar}}}$:

$$\begin{aligned}
 \frac{dL}{d\delta_{V_{\text{tar}}}} &= \frac{dL}{d\tilde{V}_{\text{tar}}} \underbrace{\frac{d\tilde{V}_{\text{tar}}}{d\delta_{V_{\text{tar}}}}}_{=1} \\
 &= \frac{\partial L}{\partial \tilde{Y}_{\text{tar}}} \frac{d\tilde{Y}_{\text{tar}}}{d\tilde{V}_{\text{tar}}}.
 \end{aligned} \tag{29}$$

Again, the gradients in the last line depend on the loss function $\mathcal{L}$ and dynamic model Φ .

B Relationship between Perturbation Magnitude and Impact

To illustrate the relationship between the displacement magnitude and the impact on model performance, we consider only those attacks that are actually meaningful (i.e., from Table 3, we consider those attacks where future trajectories were constrained). For those, the relationship

between ADE and the perturbation magnitude (both $D_{\max}$ and D_{mean}) seem to generally follow a roughly linear relationship (Figure 5). Specifically, on the *L-GAP* dataset, the coefficients of determination between the maximum/mean perturbation magnitude and ADE are $R_{\max}^2 = 0.986$ and $R_{\text{mean}}^2 = 0.962$, respectively. For the *highD* dataset, meanwhile, $R_{\max}^2 = 0.781$ and $R_{\text{mean}}^2 = 0.705$. The linear correlation is less pronounced here than on *L-GAP*; due to the longer prediction horizon, the overall model performance is worse, and correspondingly, there is more room for variance in the ADE values.

Lastly, on the *round* dataset, $R_{\max}^2 = 0.777$ and $R_{\text{mean}}^2 = 0.766$. While those values are again lower than in the ones from *L-GAP* (mostly because we also had the significant influence of the attacked model), they are still significant. This – together with Section 6.6 – strengthens the assumption that the allowed maximum displacement $d_{\max}$ is likely the deciding factor in determining the impact of the adversarial attack on the model performance. However, as increasing this makes the connection to the ground truth future weaker, a certain limit is still of utmost importance.

C Online Resources

The Supplementary Materials with further example figures (some referenced here) can be found in our github page: https://github.com/julianschumann/General-Framework-Adversarial-Attack/blob/main/Supplementary_Material%20-%20Examples.pdf.

D Examples of Generated Attacks

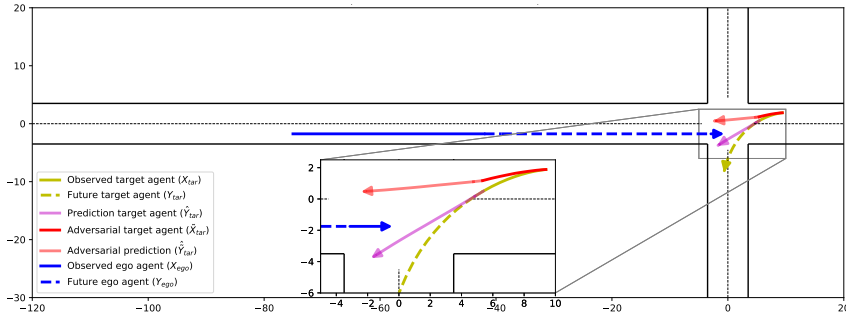


Fig. 6. Perturbing *control actions* with attack loss $\mathcal{L} = l_{\text{ADE}}(Y_{\text{tar}}, \hat{Y}_{\text{tar}}) + l_{\text{Time}}(\tilde{X}_{\text{tar}}, X_{\text{tar}})$ - ADE attack with control action dynamic constraints creates a realistic trajectory, with a gradual increase in deviation from the ground truth.

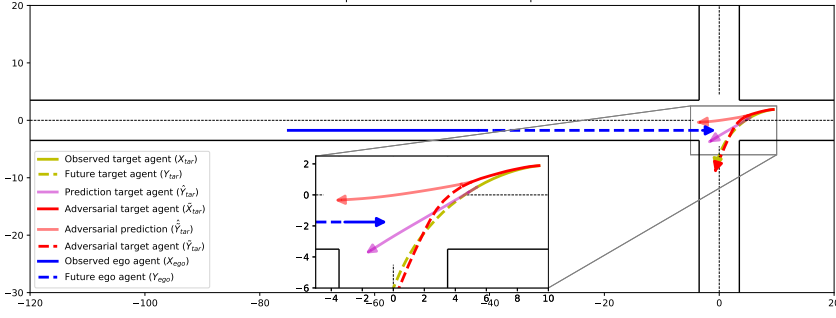


Fig. 7. Perturbing *control actions* with attack loss $\mathcal{L} = l_{ADE}(Y_{tar}, \hat{Y}_{tar}) + l_{Time}(\tilde{X}_{tar}, X_{tar}) + l_{Traj}(\tilde{Y}_{tar}, Y_{tar})$ - Feasible ADE attack with control action dynamic constraints creates a realistic trajectory, with a gradual increase in deviation from the ground truth. The adversarial observed states smoothly transition to the adversarial future states that closely resemble the future ground truth states.

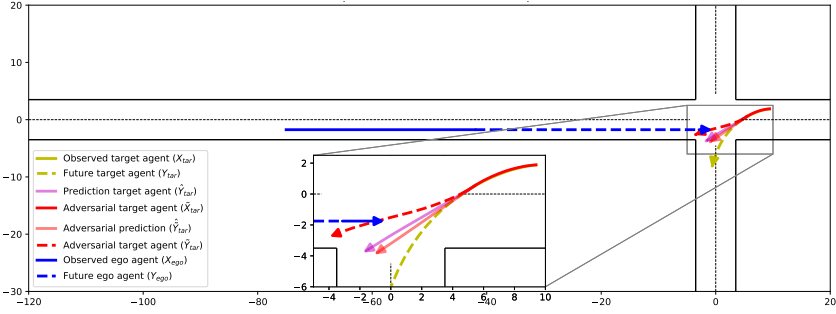


Fig. 8. Perturbing *control actions* with attack loss $\mathcal{L} = l_{FNC}(Y_{ego}, \tilde{Y}_{tar}, \hat{Y}_{tar}) + l_{Time}(\tilde{X}_{tar}, X_{tar})$ The prediction on the adversarial observed states is directed toward the nominal setting prediction. The perturbed future states are directed toward the ego agent, causing a collision.

Received 12 May 2025; revised 26 January 2026; accepted 19 February 2026