On Adversarial Robustness of Trajectory Prediction for Autonomous Vehicles

Qingzhao Zhang\textsuperscript{1}, Shengtuo Hu\textsuperscript{1}, Jiachen Sun\textsuperscript{1}, Qi Alfred Chen\textsuperscript{2}, Z. Morley Mao\textsuperscript{1}

\textsuperscript{1}University of Michigan, \textsuperscript{2}University of California, Irvine

\{qzzhang, shengtuo, jiachens, zmao\}@umich.edu  alfchen@uci.edu

Abstract

Trajectory prediction is a critical component for autonomous vehicles (AVs) to perform safe planning and navigation. However, few studies have analyzed the adversarial robustness of trajectory prediction or investigated whether the worst-case prediction can still lead to safe planning. To bridge this gap, we study the adversarial robustness of trajectory prediction models by proposing a new adversarial attack that perturbs normal vehicle trajectories to maximize the prediction error. Our experiments on three models and three datasets show that the adversarial prediction increases the prediction error by more than 150%. Our case studies show that if an adversary drives a vehicle close to the target AV following the adversarial trajectory, the AV may make an inaccurate prediction and even make unsafe driving decisions. We also explore possible mitigation techniques via data augmentation and trajectory smoothing.

1. Introduction

Autonomous vehicles (AVs) are transforming the transportation systems. AV is a complex system integrating a pipeline of modules such as perception of obstacles, planning of driving behaviors, and controlling of the physical vehicle [1, 3]. Specifically, trajectory prediction in the perception module predicts the future trajectories of nearby moving objects. The prediction is essential for the planning module and affects AV’s driving behavior. Therefore, accurate trajectory prediction is critical for safe AV driving.

Many studies propose trajectory prediction models based on deep neural networks [8, 10, 11, 13, 21, 23, 24, 27, 32, 33, 35, 38, 41, 42]. They evaluate the models on benchmarks collected from real world using the average $\ell_2$ distance between ground truth and predicted trajectories as the key metric. However, few studies evaluate trajectory prediction models from the perspective of security or analyze the robustness against adversarial examples. For trajectory prediction, if the adversary can control the position of a vehicle close to the target AV, e.g., by driving the vehicle along a crafted trajectory, the adversary can influence the AV’s trajectory prediction and driving behaviors.

To bridge this gap, we propose new white/black box adversarial attacks on trajectory prediction, which adds minor perturbation on normal trajectories to maximize the prediction error. Compared with adversarial attacks on image/video classification, attacking trajectory prediction is unique in two aspects. First, the attack requires naturalness [40] of the adversarial examples. Adversarial trajectories are natural if they obey physical rules and are possible to happen in the real world. With naturalness, the trajectories can be reproduced by the attacker-controlled vehicle in the real world and cannot be easily classified as anomaly by AVs. To realize naturalness, we enforce constraints on physical properties (e.g., velocity and acceleration) of the perturbed trajectory during optimization solving. Second, we need to define optimization objectives that are semantically-attractive for attackers targeting trajectory prediction. To this end, we find multiple attractive attack dimensions can co-exist even for the same scenario (e.g., causing the predicted trajectory to deviate laterally or longitudinally are both of interest to attackers in AV context). Thus, in our attack design we consider different metrics of prediction error as optimization objectives, e.g., average lateral/longitudinal deviation to four different directions.

We evaluate the proposed attacks on 10 different combinations of prediction models [18, 20, 30] and trajectory datasets [2, 5, 16]. The results show that the adversarial perturbation can substantially increase the prediction error by around 150%. 62.2% of attacks cause prediction to deviate by more than half of the lane width, which are likely to significantly alter AV’s navigation decisions. In addition, we thoroughly analyze how various factors impact the attack results and make recommendations for improvements of implementing trajectory prediction such as leveraging map information and driving rules. We also explore mitigation mechanisms against adversarial trajectories through data augmentation and trajectory smoothing, which reduce the prediction error under attacks by 28%. In general, our work exposes the necessity of evaluating the worst-case performance of trajectory prediction. The hard cases involving natural but adversarial trajectories generated by attacks have critical safety concerns (e.g., causing hard brakes or even...
collisions) as demonstrated by our case studies. Our main contributions are summarized as follows:

- We propose the first adversarial attack and adversarial robustness analysis on trajectory prediction for AVs considering real-world constraints and impacts.
- We report a thorough evaluation of adversarial attacks on various prediction models and trajectory datasets.
- We explore mitigation methods against adversarial examples via data augmentation and trajectory smoothing.

2. Background and Related Work

Autonomous vehicles. The autonomous vehicle is a cyber-physical system where the perception module learns the surrounding environment through sensors, the planning module makes decisions on the driving behaviors, and the control module physically operates the vehicle. Within this pipeline, trajectory prediction is required by AV systems (e.g., Baidu Apollo [3], Autoware [1]) as a part of the perception module. It predicts the future trajectories of nearby moving objects such as vehicles and pedestrians, which are important inputs for the planning module. Hence, accurate trajectory prediction is critical for safe AV driving.

Trajectory prediction models. Trajectory prediction, for AVs especially, is the problem of predicting future spatial coordinates of various road agents such as vehicles and pedestrians. These models are usually deep neural networks accepting spatial coordinates of observable road agents in the past a few seconds as the primary input and may extend the input by leveraging auxiliary features (e.g., vehicles’ heading) [11, 20], interaction among road agents [10, 18, 42], physical dynamics [30], or semantic maps [13, 25, 27, 30] to improve the accuracy of prediction. The existing evaluation metrics include Average Displacement Error (ADE), Final Displacement Error (FDE) [9], off-road rate [5], etc. These metrics reflect the average performance of trajectory prediction in the testing datasets. Instead, we focus on adversarial robustness and the worst-case performance of trajectory prediction algorithms.

Adversarial robustness. Since deep learning models are generally vulnerable to adversarial examples, various studies analyze the adversarial robustness of neural networks [7, 12, 14, 37, 39]. In AV systems, studies show that tasks such as object detection [6, 36], object tracking [17], and lane detection [31] can be affected by perturbing sensor signals or adding physical patches. However, no existing work studied the adversarial robustness of trajectory prediction.

Testing of trajectory prediction. AdvSim [15] leverages adversarial machine learning (AML) to generate high-risk driving scenarios for end-to-end testing AV systems. Instead, we aim to understand the vulnerabilities in prediction algorithms specifically. Saadatnejad et. al. [29] analyzes the inaccuracy of attention mechanisms in trajectories prediction algorithms but does not consider safety impact on real-world applications. We are the first to bridge adversarial robustness and real-world systems like AVs.

3. Problem Formulation

In this section, we first introduce the formulation of the trajectory prediction task (§3.1). We then propose the attack model (§3.2), metrics of attack impact (§3.3).

3.1. Trajectory Prediction

In this work, we focus on trajectory prediction which is executed repeatedly at a fixed time interval and makes one prediction at each time frame according to the current/history state of all observable objects (i.e., vehicles/peDESTrians). First, we denote the state of an object $i$ at time $t$ as $s^i_t$, including information of spatial coordinates and other optional features. The trajectory of object $i$ is a sequence of object states from time frame $t_1$ to $t_2$ (inclusive), denoted as $s^i_{t_1:t_2} = \{s^i_{t_1}, \ldots, s^i_{t_2}\}$. At each time frame, the prediction algorithm consumes the history trajectories of objects to predict their future trajectories, which are optimized towards having the same distribution as the ground-truth future trajectories. At time frame $t$, we denote the number of observed objects as $N$ and denote the number of time frames in history and future trajectories as $L_H$ and $L_O$ respectively. Then history trajectories are $H_t = \{H^i_t = s^i_{t-L_H+1:t}\mid i \in [1, N]\}$; ground-truth future trajectories are $F_t = \{F^i_t = s^i_{t+1:t+L_O}\mid i \in [1, N]\}$; predicted trajectories are $P_t = \{P^i_t = p^i_{t+1:t+L_O}\mid i \in [1, N]\}$ ($p^i_t$ is the predicted state of object $i$ at time $t$).

3.2. Attack Model

In this paper, we focus on the setting where the adversary drives one vehicle called “the other vehicle” (OV) along a crafted trajectory. The AV observes the OV and applies iterative trajectory prediction which produces the predicted trajectory of the OV at each time frame. The adversary controls the OV’s whole trajectory to maximize the prediction error or make the AV take unsafe driving behaviors. Figure 1 demonstrates one example of the attack. By driving along a crafted trajectory, the OV seems like changing its
lane in the AV’s prediction while the OV is actually driving straightly. Given the high-error prediction, the AV takes brake to yield the OV. If the AV brakes on the highway, it is a serious safety hazard that may cause rear-end collisions.

In a realistic attack scenario, the adversary needs to access all parameters or only APIs of the prediction model equipped by the victim AV for white-box and black-box settings respectively. When the adversary drives the OV close to the AV and is ready to attack, he/she first selects a future period and predicts the trajectories of surrounding on-road objects in that period, which are necessary inputs for generating the adversarial trajectory. The adversary then computes the adversarial trajectory for that future period and drives exactly on it (e.g., the adversary can control the OV using software). Though the generation of adversarial trajectories relies on predicted future knowledge that is not guaranteed to be accurate, the attack is still effective since the attack impact is mostly determined by the OV’s trajectory itself. The experimental evidences are in the appendix.

In addition, adversarial trajectories must satisfy the following requirements to ensure naturalness. First, the trajectory obeys the physical rules. The physical properties (e.g., velocity, acceleration) must be bounded so that real vehicles can reproduce the trajectory. Second, the trajectory represents normal driving behavior instead of ruthless driving.

### 3.3. Evaluation Metrics

We use six metrics to evaluate the prediction error. Also, an attack is effective if the prediction error is significantly raised after the adversarial perturbation. We first include two metrics that are commonly used in related works [9]. (1) Average displacement error. The average of the root mean squared error (RMSE) between the predicted and ground-truth trajectory. (2) Final displacement error. The RMSE between the predicted and ground-truth trajectory position at the last predicted time frame.

However, the above two metrics are not sufficient to evaluate the impact of targeted attacks. For instance, to spoof lane-changing behavior to left (Figure 1), the predicted trajectory should have a deviation to left. Similarly, to spoof a fake acceleration, the deviation should be towards the front of the longitudinal direction. For the above attacks, only the deviation to a specific direction is counted as an effective attack impact. Therefore, we design four extra metrics including the average deviation towards left/right side of lateral direction and front/rear side of longitudinal direction. The metrics are formally defined as:

\[
D(t, n, R) = \frac{1}{L_O} \sum_{\alpha=t+1}^{t+L_O} (p_{\alpha,n} - s_{\alpha,n})^T \cdot R (s_{\alpha+1,n}, s_{\alpha,n}),
\]

where \( t \) denotes the time frame ID, \( n \) denotes the target vehicle ID, \( p \) and \( s \) are binary vectors representing predicted and ground-truth vehicle locations respectively, and \( R \) is a function generating the unit vector to a specific direction. We approximate longitudinal direction as \( s_{\alpha+1,n} - s_{\alpha,n}. \)

According to our case studies, we believe that half of the lane width (e.g., about 1.85 meters in datasets we use) is a threshold of deviation to cause real-world impacts in a high probability. If the average deviation exceeds the threshold, the last predicted trajectory location is likely on a different lane, which may lead to different decisions of the AV.

### 4. Adversarial Example Generation

We design white box and black box attacks against trajectory prediction following definitions in §3.1. **Perturbation.** We generate the adversarial trajectory by adding minor perturbation on normal trajectories. The perturbation changes the spatial coordinates (x-y location) and features that are calculated from locations (e.g., velocity) correspondingly. As shown in Figure 1, perturbation is applied on the history trajectory in length of \( L_O \) to control the prediction. Since only the prediction of the current time frame is considered, we name this attack single-frame attack. However, real-world safety issues usually happen in a longer time sequence. Therefore, we generalize the attack to multi-frame, as shown in Figure 2. We define parameter \( L_P \) to represent the number of predictions considered in the attack objective. Given \( L_P \), one attack scenario include \( L_I + L_O + L_P - 1 \) time frames. We apply perturbation on the first \( L_I + L_P - 1 \) time frames so that predicted trajectories on time frames \( \{L_I, L_I + 1, \ldots, L_I + L_P - 1\} \) (\( L_P \) frames in total) are controlled by adversarial trajectories. We maximize the total prediction error in these \( L_P \) time frames to launch the multi-frame attack.

**Objective.** The optimization has six different objective functions corresponding to the six metrics of prediction error. The objective is the negation of the average prediction error in all considered time frames:

\[
L(n, f) = -\frac{1}{L_P} \sum_{\alpha=L_I}^{L_I+L_P-1} f(P_{\alpha}^n, F_{\alpha}^n),
\]

where \( f \) denotes the function calculating one of the six metrics; \( n \) is ID of the target vehicle (i.e., OV); \( P \) and \( F \) represents predicted and future trajectories (§3.1).

**Hard constraints of perturbation.** As mentioned in §3.2, the perturbed trajectories must be physically feasible and not perform dangerous driving behaviors. To enforce this
requirement, we design constraints that need to be satisfied by any perturbation.

First, the bounds of physical properties. We traverse all trajectories in the testing dataset to calculate the mean ($\mu$) and standard deviation ($\sigma$) of (1) scalar velocity, (2) longitudinal/lateral acceleration, and (3) derivative of longitudinal/lateral acceleration. For each $\mu$ and $\sigma$, we check that the value of perturbed trajectories does not exceed $\mu \pm 3\sigma$. Assuming the physical properties are in the normal distribution, this range covers 99.9% of the dataset but excludes outliers. We also manually check that the bounds are physically reasonable. In addition, when checking the physical constraints, three ground-truth trajectory points are involved before and after the perturbed part thus the boundary of normal and perturbed trajectories is natural.

Second, the bound of deviation on each trajectory location. We set the maximum deviation as 1 meter by default. Given that the urban lane width is around 3.7 meters in datasets we use and the width of cars is about 1.7 meters, the 1-meter deviation is an upper bound for a car not shifting to another lane if it is normally driving in the center of the lane. The bound (1) rules out ruthless driving by preserving the original driving behavior and (2) is a parameter for tuning the stealth level of the attack.

We enforce the constraints by reducing the perturbation whenever the constraints are violated. Given the perturbation $\Delta$, the history trajectory of the target vehicle $H_n$, and the constraint function $C$, we calculate the maximum coefficient $0 \leq \theta \leq 1$ which reduces the perturbation to $\theta\Delta$ while $\theta\Delta$ satisfies all constraints. Formally speaking, the calculation of $\theta$ is an optimization problem:

$$\max \theta \quad \text{s.t.} \quad C(H^n + \theta\Delta) \land 0 \leq \theta \leq 1$$

(3)

White box optimization. We design our white box optimization method based on Projected Gradient Descent (PGD) [22]. The process can be summarized as follows.

The perturbation is initialized randomly. In each iteration, we first enforce the hard constraint on the current perturbation following Equation 3 and then add the perturbation on the original history trajectories of the target vehicle $n$ which is observed by the AV. Then $L_P$ times of prediction is executed on the perturbed trajectory data and the loss of the iteration is calculated using Equation 2. Next, the algorithm updates the perturbation using gradient descents. Finally, the algorithm produces the best perturbation which can transform the original scenario to the worst-case scenario with the maximized prediction error.

Black box optimization. Methods based on gradient descent are not always feasible because the trajectory prediction models may have non-differentiable layers. Hence, we design a black box attack method based on Particle Swarm Optimization (PSO) [19] which requires only the API of model inference instead of gradients. PSO is an optimization method by iteratively improving solution candidates (i.e., particles) with regards to a given measure of quality in the search space. In this case, each particle is one candidate of perturbation, the measure of quality is defined by objective function (Equation 2), and the search space is defined by the hard constraints (Equation 3).

5. Mitigation Mechanisms

From our observation of attack results (§6), the adversarial trajectories frequently change acceleration, which is a rare pattern in normal trajectories. Based on this phenomenon, we design mitigation methods as follows.

Data augmentation. Since normal trajectories in the training dataset are mostly smooth with stable acceleration, adversarial trajectories have a different data distribution. Hence, we apply data augmentation to inject adversarial patterns in the training data. During the training, we add random perturbation on randomly selected trajectories while the perturbation satisfies the hard constraints defined in §4. We do not adopt adversarial training because of its limitations such as the high cost of training and poor generality in terms of attack goals.

Train-time trajectory smoothing. Since the unstable velocity or acceleration is a key pattern of adversarial trajectories, we can partially remove the adversarial effect by smoothing the trajectories. We apply trajectory smoothing on both training and testing data. There are various choices of smoothing algorithms and we use a simple linear smoother based on convolution in our experiments. This mitigation relies on the physical properties of trajectories instead of gradient obfuscation [4]. Hence, it does not matter if the attacker knows the gradients of the smoothing.

Test-time detection and trajectory smoothing. The above two mitigation methods modify the distribution of training data so that one needs to retrain the model. To make the mitigation easier to deploy, we propose another method that only smooths trajectories during inference time if the trajectory is detected as adversarial. We design two methods for detecting adversarial trajectories. First, SVM classifier [34]. We extract the magnitude and direction of the acceleration as features to fit an SVM model for classifying normal and adversarial trajectories. Second, rule-based detector. We calculate the variance of acceleration over time frames and if the variance is higher than a threshold the trajectory is detected as adversarial.

6. Experiments

In this section, we evaluate proposed attack/mitigation methods and analyze the results.
6.1. Experimental Setting

**Datasets.** We summarize the characteristics of the three datasets we use in Table 1. We select history trajectory length \(L_I\) and future trajectory length \(L_O\) following the recommendation from the dataset’s authors. We randomly select 100 scenarios as test cases from each dataset.

**Models.** We summarize three prediction models we use in Table 2, which are open-source state-of-the-art models at the time of the experiments. Note that Trajectron++ model produces multiple predicted trajectories with probabilities (i.e., multi-prediction) and we select the trajectory with the highest probability as the final result. Also, Trajectron++ requires the semantic map as input which is only available in nuScenes dataset. Therefore, we prepare two versions of Trajectron++. Trajectron++ (w/o map) is evaluated on all datasets while Trajectron++ (w/ map) is evaluated on nuScenes. For each combination of models and datasets, we train the model using fine-tuned hyperparameters.

**Implementation details.** Our implementation is open sourced at https://github.com/zqzqz/AdvTrajectoryPrediction. For the PGD-based white box attack, we use Adam optimizer with a learning rate of 0.01 and set the maximum iteration to 100. For PSO-based black box attack, we set the number of particles to 10, inertia weight to 1.0, acceleration coefficients to (0.5, 0.3), and the maximum iteration to 100. For mitigation, we use convolution-based linear smoother for trajectory smoothing and the SVM implementation in scikit-learn [26] for anomaly detection. More details are in the appendix.

6.2. Attack Results

**General results.** For each model & dataset combination, each test case, and each perturbation objective, we first execute the white-box attack in the setting of single-frame prediction \((L_P = 1)\). We assume the attacker acknowledges history trajectories of all objects on-road and discuss a more realistic setting in the appendix. We present the average prediction error before and after perturbation in Table 3. Generally, the white-box adversarial perturbation is effective on all models and datasets. On average, ADE/FDE is increased by 167%/150%. The lateral/longitude deviation reaches 2.03/3.84 meters and 62.2% of attacks are likely to cause real-world impact since their average deviations are larger than half of the lane width (i.e., 1.85 meters).

**Naturalness.** The adversarial trajectories can naturally happen in the real world in a reasonable probability. One characteristic of adversarial trajectories is that they often change vehicle velocity or acceleration. However, a small fraction of normal trajectories share the same pattern and are indistinguishable from adversarial trajectories. The hard constraint (§4) also ensures that adversarial trajectories are physically feasible to reproduce by driving a real car. Hence, the adversarial attack can be regarded as a method to discover the worst but realistic cases of prediction.

Next, we will analyze factors that affect adversarial robustness in §6.2.1, §6.2.2, and §6.2.3. The analysis uses experiments on dataset Apolloscape as supporting evidence. As a baseline, in the single-frame white-box attack on Apolloscape, ADE/FDE is increased by 239%/206% while lateral/longitude deviation reaches 2.49/6.31 meters on average (53.6% of deviations are larger than 1.85 meters). If not specified, the percentage rise/drop of prediction error is the average of six metrics.

6.2.1 Different Scenarios

**High-acceleration scenarios.** The prediction error is generally high in scenarios where vehicles have high acceleration. Taking metric ADE as an example, the prediction error in high-acceleration scenarios (average acceleration > 1m/s²) w/o and w/ perturbation are 31% and 20% higher than low-acceleration scenarios respectively. Typical high-acceleration scenarios include turning at intersections or stopping at stop lines (details in the appendix). All three models cannot foresee the driver’s behavior in such scenarios or stopping at stop lines (details in the appendix). All three models cannot foresee the driver’s behavior in such scenarios or stopping at stop lines (details in the appendix). All three models cannot foresee the driver’s behavior in such scenarios or stopping at stop lines (details in the appendix). All three models cannot foresee the driver’s behavior in such scenarios or stopping at stop lines (details in the appendix).

**Density of traffic.** Prediction models commonly encode the interaction among objects as a graph representation to improve prediction accuracy. To study the factor of traffic density, we repeat the white box attack on three models using Apolloscape dataset but drops 50% or 100% of objects other than the target vehicle. However, the prediction error under the white-box attack appears independent of traffic density. The result shows that the worst-case prediction error under attacks is mostly determined by the target vehicle’s trajectory rather than the surrounding objects.

6.2.2 Different Models

**Input representation.** Features independent of trajectory locations (e.g., maps) help to improve adversarial robustness. Comparing Trajectron++ w/ map against w/o map, ADE on original and perturbed trajectories are reduced by 22% and 31% respectively. By adding map information as input features, the weight on perturbed features is lower so that the prediction result is less sensitive to the perturbation.
Table 3. Average prediction error before and after single-frame black box attack.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>ADE (m)</th>
<th>FDE (m)</th>
<th>Left (m)</th>
<th>Right (m)</th>
<th>Front (m)</th>
<th>Rear (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Normal / Attack</td>
<td>Normal / Attack</td>
<td>Normal / Attack</td>
<td>Normal / Attack</td>
<td>Normal / Attack</td>
<td>Normal / Attack</td>
</tr>
<tr>
<td>GRIP</td>
<td>ApolloScape</td>
<td>1.97 / 7.14</td>
<td>3.18 / 10.74</td>
<td>-0.0128 / 2.65</td>
<td>0.0128 / 2.38</td>
<td>-0.0154 / 4.80</td>
<td>0.0154 / 5.49</td>
</tr>
<tr>
<td>GRIP</td>
<td>NGSIM</td>
<td>0.29 / 9.24</td>
<td>1.28 / 16.5</td>
<td>-0.17 / 1.1</td>
<td>0.17 / 0.31</td>
<td>-5.3 / -3.76</td>
<td>5.3 / 3.85</td>
</tr>
<tr>
<td>GRIP</td>
<td>nuScenes</td>
<td>5.46 / 8.32</td>
<td>10.37 / 15.4</td>
<td>0.23 / 1.53</td>
<td>-0.23 / 1.44</td>
<td>-1.04 / 1.50</td>
<td>1.04 / 1.76</td>
</tr>
<tr>
<td>FQA</td>
<td>ApolloScape</td>
<td>3.77 / 5.64</td>
<td>3.82 / 8.78</td>
<td>0.0479 / 2.10</td>
<td>-0.0479 / 1.86</td>
<td>-0.387 / 3.11</td>
<td>0.387 / 3.84</td>
</tr>
<tr>
<td>FQA</td>
<td>NGSIM</td>
<td>5.44 / 7.05</td>
<td>9.45 / 12.2</td>
<td>0.205 / 1.19</td>
<td>-0.205 / 0.787</td>
<td>-1.1 / 2.12</td>
<td>1.11 / 2.16</td>
</tr>
<tr>
<td>FQA</td>
<td>nuScenes</td>
<td>5.28 / 7.83</td>
<td>10.07 / 14.4</td>
<td>0.229 / 1.18</td>
<td>-0.229 / 0.736</td>
<td>-0.814 / 1.79</td>
<td>0.814 / 3.347</td>
</tr>
<tr>
<td>Trajectron++</td>
<td>ApolloScape</td>
<td>1.31 / 7.11</td>
<td>2.28 / 10.8</td>
<td>0.252 / 4.74</td>
<td>-0.252 / 3.49</td>
<td>-0.838 / 4.38</td>
<td>0.838 / 7.24</td>
</tr>
<tr>
<td>Trajectron++</td>
<td>NGSIM</td>
<td>2.31 / 9.84</td>
<td>4.99 / 16.9</td>
<td>0.219 / 4.39</td>
<td>0.219 / 4.62</td>
<td>-0.085 / 5.43</td>
<td>0.085 / 7.28</td>
</tr>
<tr>
<td>Trajectron++</td>
<td>nuScenes</td>
<td>5.31 / 8.73</td>
<td>11.2 / 17.1</td>
<td>-0.331 / 2.10</td>
<td>0.331 / 2.68</td>
<td>-1.57 / 2.25</td>
<td>1.57 / 5.01</td>
</tr>
<tr>
<td>Trajectron++(map)</td>
<td>nuScenes</td>
<td>2.67 / 6.81</td>
<td>3.39 / 11.8</td>
<td>0.107 / 2.118</td>
<td>0.107 / 2.25</td>
<td>-0.526 / 3.79</td>
<td>0.526 / 4.48</td>
</tr>
</tbody>
</table>

Figure 3. White box v.s. block box attack (single-frame) on ApolloScape dataset.

Also, good prediction accuracy on normal trajectories does not necessarily lead to good adversarial robustness. Without perturbation, Trajectron++ has better prediction accuracy on all three datasets, thanks to its comprehensive input representation. With adversarial perturbation, however, Trajectron++ does not have the best accuracy. On dataset ApolloScape and NGSIM, FQA has a better prediction error against perturbation. FQA’s single LSTM makes predictions mainly based on the last two trajectory positions so that the perturbation on other parts of trajectories is not effective. Though Trajectron++ integrates rich features such as dynamics, the features are affected by perturbation on any trajectory position, which is a wide attack surface.

6.2.3 Different Attack Methods

White box v.s. black box. We conduct black box attacks on the three models and evaluate the attack result on the ApolloScape dataset. We visualize the six metrics of prediction error in Figure 3. In general, black box attacks and white box attacks have a very similar performance. Since the search space of the optimal perturbation is in a two-dimensional space (i.e., the spatial locations) and restricted by hard constraints (§4), the attacker can efficiently solve the optimization problem without knowledge of the model. White-box and black-box adversarial trajectories both have high variance of acceleration but perturbation of white-box attack is overall smaller because the guidance of gradient is limited by hard constraints. Also, good prediction accuracy on normal trajectories does not necessarily lead to good adversarial robustness. Without perturbation, Trajectron++ has better prediction accuracy on all three datasets, thanks to its comprehensive input representation. With adversarial perturbation, however, Trajectron++ does not have the best accuracy. On dataset ApolloScape and NGSIM, FQA has a better prediction error against perturbation. FQA’s single LSTM makes predictions mainly based on the last two trajectory positions so that the perturbation on other parts of trajectories is not effective. Though Trajectron++ integrates rich features such as dynamics, the features are affected by perturbation on any trajectory position, which is a wide attack surface.

Table 4. Attack transferability (white box, single-frame).

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>FQA</th>
<th>GRIP++</th>
<th>Trajectron++</th>
</tr>
</thead>
<tbody>
<tr>
<td>FQA</td>
<td>100%</td>
<td>42.8%</td>
<td>16.5%</td>
<td></td>
</tr>
<tr>
<td>GRIP++</td>
<td>60.8%</td>
<td>100%</td>
<td>15.5%</td>
<td></td>
</tr>
<tr>
<td>Trajectron++</td>
<td>64.7%</td>
<td>39.5%</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Attack transferability (white box, single-frame).

Figure 3. White box v.s. block box attack (single-frame) on ApolloScape dataset.

Table 4. Attack transferability (white box, single-frame).

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>FQA</th>
<th>GRIP++</th>
<th>Trajectron++</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100%</td>
<td>42.8%</td>
<td>16.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60.8%</td>
<td>100%</td>
<td>15.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64.7%</td>
<td>39.5%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Figure 3. White box v.s. block box attack (single-frame) on ApolloScape dataset.

Table 4. Attack transferability (white box, single-frame).

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>FQA</th>
<th>GRIP++</th>
<th>Trajectron++</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100%</td>
<td>42.8%</td>
<td>16.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60.8%</td>
<td>100%</td>
<td>15.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64.7%</td>
<td>39.5%</td>
<td>100%</td>
</tr>
</tbody>
</table>
6.2.4 Transferability

We evaluate the transferability across the three models using Apolloscape dataset. Since the attack impact is evaluated using quantified prediction errors instead of the binary judgment of attack success, we define a percentage score to measure the transferability. When we apply the perturbation optimized on the source model to the target model, for each metric of prediction error, we calculate the ratio of prediction error on the target model to prediction error on the source model. Finally, we take the average of six ratio numbers corresponding to six metrics as the final score. The results are shown in Table 4. First of all, 90.25% of transferred adversarial trajectories successfully increase the prediction error compared with non-attack cases. It shows that the perturbation exploits common internal patterns of general prediction models. Therefore, the perturbation optimized on one model can launch attacks on other models as well. It means the perturbation brings common patterns causing deviation of the predicted trajectory. Second, the transferability is highly correlated with the target model: transferring the perturbation to FQA is easier but to Trajectron++ is harder. We hypothesize that it is because Trajectron++ leverages more features of trajectories so that the perturbation optimized on fewer features cannot completely reproduce the high prediction error on Trajectron++.

6.3. Mitigation Results

We test the mitigation mechanisms using the three models and Apolloscape dataset. The result is shown in Figure 6. We assume that the attacker has full knowledge of the mitigation method and applies the same mitigation method on each prediction during the white-box attack. Our convolution-based trajectory smoothing is differentiable so that the computed gradient can directly involve the effect of the mitigation. If the smoothing method is replaced by non-differentiable ones, the attacker can also approximate the gradient using a differentiable function given the full knowledge of the smoothing algorithm.

Train-time mitigation. The effectiveness of data augmentation and trajectory smoothing varies on different models. Data augmentation is effective on Trajectron++ (prediction error reduced by 24%), which deploys a complicated network structure on low-dimensional data (i.e., trajectories). Data augmentation can alleviate the over-fitting issues of Trajectron++. Trajectory smoothing is effective on FQA. FQA model has potential under-fitting issues since its predicted trajectories mainly depend on the direction of velocity at the last time frame and have a high error on curve trajectories. Trajectory smoothing cannot solve the problem of the model but directly alleviate the impact of adversarial perturbation on the last two observed trajectory positions. If we apply data augmentation and trajectory smoothing at train time simultaneously, the prediction error under attacks is reduced by 26% on average while the prediction error of normal cases is increased by 11%.

Test-time mitigation. If applying trajectory smoothing on all trajectories, the prediction error under attacks is dropped by 13% but the normal-case prediction error raised significantly by 28%. This is because the testing data distribution is altered to be different from the training dataset. To address the problem, we require the detection method to distinguish adversarial trajectories from normal ones and apply smoothing only on adversarial examples.

Figure 7 shows the ROC curve of two detectors mentioned in §5. First, our rule-based method (i.e., the threshold on the variance of acceleration) has better performance than SVM classification in terms of true positive rate (TPR) and false positive rate (FPR). This result confirms that the variance of acceleration over time is the key difference between adversarial and normal trajectories. Second, the de-
In this section, we demonstrate the real-world impact of the adversarial perturbation by one case study. More case studies are in the appendix.

6.4. Case study

In this section, we demonstrate the real-world impact of the adversarial perturbation by one case study. More case studies are in the appendix.

7. Conclusion

We present the first effort of analyzing adversarial robustness of trajectory prediction. From the evaluation of our proposed attack, prediction models are generally vulnerable to adversarial perturbation and may cause dangerous AV behavior such as hard brakes. We shed light on the necessity of evaluating worst-case prediction accuracy under hard scenarios or adversarial examples. To improve adversarial robustness of trajectory prediction, we propose several mitigation methods. We also suggest leveraging map information and semantic of driving rules to guide prediction.

Acknowledgments. This work was supported by NSF under the grant CNS-1930041, CNS-1932464, CNS-1929771, CNS-2145493 and the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant #2112562.
References


