Robust Trajectory Prediction against Adversarial Attacks

Anonymous Author(s)
Affiliation
Address
email

Abstract: Trajectory prediction using deep neural networks (DNNs) is an essential component of autonomous driving (AD) systems. However, these methods are vulnerable to adversarial attacks, leading to serious consequences such as collisions. In this work, we identify two key ingredients to defend trajectory prediction models against adversarial attacks including (1) designing effective adversarial training methods and (2) adding domain-specific data augmentation to mitigate the performance degradation on clean data. We demonstrate that our method is able to improve the performance by 46% on adversarial data and at the cost of only 3% performance degradation on clean data, compared to the model trained with clean data. Additionally, compared to existing robust methods, our method can improve performance by 21% on adversarial examples and 9% on clean data. Our robust model is evaluated with a planner to study its downstream impacts. We demonstrate that our model can significantly reduce the severe accident rates (e.g., collisions and off-road driving).

1 Introduction

Trajectory prediction is a critical component of modern autonomous driving (AD) systems. It allows an AD system to anticipate the future behaviors of other nearby road participants and plan its actions accordingly. Recent trajectory prediction models built on Deep Neural Networks (DNN) have demonstrated state-of-the-art performance on large-scale benchmarks [1–7], showing a promising path towards learning-based trajectory prediction for AD systems. As trajectory prediction plays an important role in AD systems, accurate predictions are required for making safe driving decisions. It is crucial to understand how unknown scenarios will affect trajectory predictions and then bolster the robustness of such trajectory predictions in return.

To achieve this goal, adversarial attacks [8–10] are often used as a proxy to measure the worst-case performance of the model when facing unseen scenarios. Similarly, we use a standard adversarial attack setup [11] for trajectory predictions. As illustrated in Fig. 1, an adversarial agent (red vehicle) aims to cause a traffic accident. It drives along a carefully designed trajectory (i.e., adv history) to influence the trajectory prediction model of the Autonomous Vehicle (green vehicle). Such an adversary can critically compromise the predicted trajectories of all other agents by altering its route in inconspicuous ways. By fooling the trajectory prediction models, it can further affect downstream planning of the AV systems and cause serious consequences. Using the adversarial attack as the proxy, this work aims to develop effective techniques to bolster the robustness of trajectory prediction models against adversarial attacks and improve the AD's safety under uncertain scenarios.

At the same time, adversarial robustness for machine learning is a widely-studied area, but most works focus on classification tasks [12–23]. Among the proposed techniques, adversarial training [8] remains the most effective and widely used method to defend classifiers against adversarial attacks. The general strategy of adversarial training is to solve a min-max game by generating adversarial examples for a model at each training step and then optimizing the model to make correct predictions

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

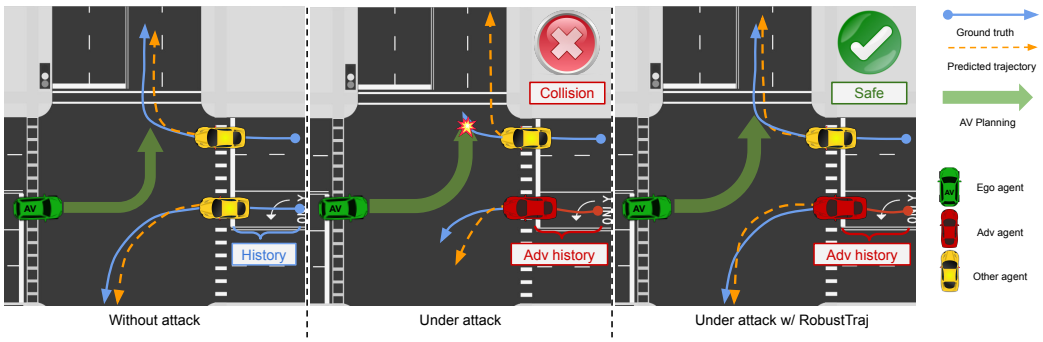


Figure 1: Overview of *RobustTraj* preventing Autonomous Vehicle (AV) from collisions when its trajectory prediction model is under adversarial attacks. When the trajectory prediction model is under attack, the AV predicts the wrong future trajectory of the other agent turning right (yellow vehicle). This results in AV speeding up instead of slowing down, and eventually colliding into the other vehicle.

for these samples. However, directly applying adversarial training to trajectory prediction presents a number of critical technical challenges.

First, most trajectory prediction methods employ probabilistic generative models to cope with the uncertainty in motion forecasting [2–7]. As we will show in this paper, the stochastic components of these models (e.g., posterior sampling in VAEs) can obfuscate the gradients that guide the adversarial generation, making naïve adversarial training methods ineffective. Second, adversarial training on trajectory prediction task aims to model joint data distribution of future trajectories and adversarial past trajectories. However, the co-evolution of the adversarial sample distribution and the prediction model during the training process makes the joint distribution hard to model and often destabilizes the adversarial training. Finally, prior work [23] shows that adversarial training often leads to degraded performance on clean (unperturbed) data, while retaining good performance in benign cases is crucial due to the critical role of trajectory prediction for AVs. Hence, an effective adversarial training method must carefully balance the benign and the adversarial performance of a model.

Our approach. We propose an adversarial training framework for trajectory predictions named *RobustTraj*, by addressing the aforementioned challenges. First, to address the issue of an obfuscated gradient in adversarial generation due to stochastic components, we devise a *deterministic attack* that creates a deterministic gradient path within a probabilistic model to generate adversarial samples. Second, to address the challenge of an unstable training process due to shift in adversarial distributions, we introduce a hybrid objective that interleaves the adversarial training and learning from clean data to anchor the model output on stable clean data distribution. Finally, to achieve balanced performances on both adversarial and clean data, we introduce a domain-specific data augmentation technique for trajectory prediction via a dynamic model. This data augmentation technique generates diverse, realistic, and dynamically-feasible samples for training and achieves a better performance trade-off on clean and adversarial data.

We empirically show that *RobustTraj* can effectively defend two different types of probabilistic trajectory prediction models [4, 7] against adversarial attacks, while incurring minimal performance degradation on clean data. For instance, *RobustTraj* can increase the adversarial performance of AgentFormer [4], a state-of-the-art trajectory prediction model, by 46% at the cost of 3% performance drop on clean data. To further show impacts of our method on the AD stack, we plug our robust trajectory prediction model into a planner and demonstrate that our model reduces serious accidents rates (e.g., collisions and off-road driving) under attacks by 100%, compared to the standard non-robust model trained using only clean data.

2 Related Work

Adversarial attacks and defenses on trajectory prediction. A recent work began to study the adversarial robustness of trajectory prediction models [11]. Zhang et al. [11] demonstrated that perturbing agents' observed trajectory can adversarially impact the prediction accuracy of a DNN-based trajectory forecasting model. To mitigate the issue, Zhang et al. [11] proposed several defense methods such as data augmentation and trajectory smoothing. However, these methods are less effective when facing adaptive attacks [24]. In our work, we propose to use adversarial training which provides the general adversarial robustness that can resist adaptive attacks.

Adversarial training. A variety of adversarial training methods have been proposed to defend DNN-based models against adversarial attacks [8, 19, 12–23]. The most common strategy is to design a min-max game with the inner maximization process and outer minimization process. The inner maximization process generates adversarial examples that maximize an adversarial objective (e.g., make wrong prediction). The outer minimization process then updates the model parameters to minimize the error on the adversarial examples. Although there exists a large body of literature in studying adversarial robustness for machine learning, most focus on the problem of discriminative model (e.g., object recognition), leaving other problem domains (e.g., conditional generative models) largely unexplored. In this work, we develop a novel adversarial training method for trajectory prediction models, where most state-of-the-art trajectory prediction models are generative and probabilistic, by addressing a number of critical technical challenges.

3 Preliminaries and Formulation

Trajectory prediction. The goal is to predict future trajectory distribution $\mathcal{P}_\theta(Y|X)$ of N agents conditioned on their H history time states $\mathbf{X} = \left(\mathbf{X}^{-H+1}, \ldots, \mathbf{X}^0\right)$, and other environment context such as maps [1] to predict T time-step future trajectories $\mathbf{Y} = \left(\mathbf{Y}^1, \ldots, \mathbf{Y}^T\right)$. For observed time steps $t \leq 0$, we denote the agent states as $\mathbf{X}^t = (x_1^t, \ldots, x_i^t, \ldots, x_N^t)$, where x_i^t is the state of agent i at the time step t. Similarly, $\mathbf{Y}^t = (y_1^t, \ldots, y_N^t)$ denotes the states of N agents at a future time step t ($t > 0$). We denote the ground truth and the predicted future trajectories as $\mathbf{Y}$ and $\hat{\mathbf{Y}}$, respectively. We denote the history information encoded by a function f as the decision context $\mathbf{C} = f(\mathbf{X})$.

Probabilistic trajectory prediction models. In this work, we focus on defending generative, probabilistic trajectory prediction models, as they have demonstrated superior performance in modeling uncertainty in predicting future motions [2–7]. We consider two most popular types of generative models: conditional variational encoders (CVAEs) and conditional GANs (cGANs), both can be viewed as latent variable models. We define latent variables $\mathbf{Z} = \{z_1, ..., z_i, ..., z_N\}$ where z_i represents the latent variable of the agent i. CVAE formulates the generative problem as: $p_\theta(\mathbf{Y}|\mathbf{X}) = \int p_\theta(\mathbf{Y}|\mathbf{X}, \mathbf{Z}) \cdot p_\theta(\mathbf{Z}|\mathbf{X}) d\mathbf{Z}$, where $p_\theta(\mathbf{Z}|\mathbf{X})$ is a conditional Gaussian prior ($\mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}), p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$) with mean $p_\theta^\mu(\mathbf{Z}|\mathbf{X})$ and standard deviation $p_\theta^\sigma(\mathbf{Z}|\mathbf{X})$; $p_\theta(\mathbf{Y}|\mathbf{X}, \mathbf{Z})$ is a conditional likelihood model. The model is usually trained through optimizing a negative evidence lower objective [4]:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{elbo}} + \mathcal{L}_{\text{diversity}} \\ &= -\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X})}[\log p_\theta(\mathbf{Y}|\mathbf{Z}, \mathbf{X})] + \text{KL}(q_\phi(\mathbf{Z}|\mathbf{Y}, \mathbf{X}) \parallel p_\theta(\mathbf{Z}|\mathbf{X})) + \min_k \parallel \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \parallel^2 ,\end{aligned} \tag{1}$$

where $q_\phi(\mathbf{Z}|\mathbf{Y}, \mathbf{X})$ is an approximate posterior parameterized by ϕ, $p_\theta(\mathbf{Z}|\mathbf{X})$ is a conditional Gaussian prior parameterized by θ, and $p_\theta(\mathbf{Y}|\mathbf{Z}, \mathbf{X})$ is a conditional likelihood modeling future trajectory $\mathbf{Y}$ via the latent codes $\mathbf{Z}$ and past trajectory $\mathbf{X}$. Additionally, $\mathcal{L}_{\text{diversity}} = \min_k \parallel \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \parallel^2$ is a diversity loss, which encourages the network to produce diverse samples. Given each past trajectory X, the model generates K sets of latent codes $\{\mathbf{Z}^{(1)}, \cdots, \mathbf{Z}^{(k)}, \cdots, \mathbf{Z}^{(K)}\}$ from the conditional

[1] For simplify, we ignore contextual information.

Gaussian prior $\mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}), p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$, where $\mathbf{Z}^{(k)} = \{z_1^k, \cdots, z_n^k\}$, resulting in K future trajectories $\hat{\mathbf{Y}}^{(k)}$.

Similarly, in a conditional Generative Adversarial Net (cGAN)-based model (e.g., Social-GAN [1]), it uses a loss function as follows:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{gan}} + \mathcal{L}_{\text{diversity}} \\ &= \mathbb{E}_{\mathbf{Y}\sim p_{\text{data}}}[\log D_\theta(\mathbf{Y}|\mathbf{X})] + \mathbb{E}_{\mathbf{Z}\sim p_{\text{Z}}}[\log(1 - D_\theta(G_\phi(\mathbf{Y}|\mathbf{X}, \mathbf{Z})))] + \min_k \parallel \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \parallel^2 ,\end{aligned} \tag{2}$$

where G represents the generator and D represents the discriminator. $\hat{\mathbf{Y}}^{(k)} = G(\mathbf{Y}|\mathbf{X}, \mathbf{Z}^{(k)})$ is one of the predicted trajectories in which $\mathbf{Z}^{(k)}$ is randomly sampled from $\mathcal{N}(0, 1)$.

Threat model. We follow the setup in prior work [11] and adopt an idealized threat model, where the adversary alters its observed history $\mathbf{X}$ by adding a perturbation δ bounded by the adversarial set $\mathbb{S}_p^\epsilon = \{\delta | \parallel \delta \parallel_p \leq \epsilon\}$, where ϵ is the perturbation bound. The perturbation aims to mislead the prediction $\hat{\mathbf{Y}}$. A naïve adversarial attack is to find the perturbation through an adversarial objective $\delta = \arg\max_{\delta\in\mathbb{S}}\{\min_{k\in\{1,\cdots,K\}} \parallel p_\theta(\mathbf{Y}|\mathbf{X}+\delta, \mathbf{Z}^{(k)}) - \mathbf{Y} \parallel^2\}$, where $p_\theta(\mathbf{Y}|\mathbf{X}+\delta, \mathbf{Z}^{(k)})$ is the predicted trajectory conditioned on the random variable $\mathbf{Z}^{(k)}$ and adversarial history trajectory $\mathbf{X}+\delta$.

Naïve adversarial training. Adversarial training formulates a min-max game with an inner maximization process that optimizes the perturbation δ to generate adversarial examples for misleading the model at each training iteration, and an outer minimization process that optimizes the model parameters to make correct predictions for these examples. We follow the standard adversarial training formulation [8]:

$$\min_{\theta,\phi} \max_{\delta\in\mathbb{S}} \quad \mathcal{L}_{\text{total}}(\mathbf{X}+\delta, \mathbf{Y}). \tag{3}$$

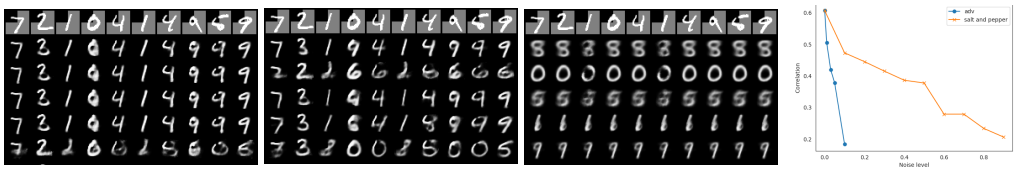
4 *RobustTraj*: Robust Trajectory Prediction

As stated earlier, applying adversarial training for trajectory prediction presents three critical challenges: (1) gradient obfuscation due to model stochasticity, (2) unstable learning due to changing adversarial distribution, and (3) performance loss in the benign situation. In this section, we describe each challenge in more detail and present the corresponding solutions in our *RobustTraj* method. **Improve adversarial generation with *Deterministic Attack*.** Since trajectory prediction is inherently uncertain and there is no single correct answer, probabilistic generative models are usually used to cope with the stochastic nature of the trajectory prediction task. Such stochasticity will obfuscate the gradients that are used to generate effective adversarial examples in the inner maximization process of adversarial training. The naïve attack mentioned in section 3 is a straightforward way to achieve this goal. However, this optimization involves a stochastic sampling process $\mathbf{Z}^{(k)} \sim \mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}),\, p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$. Such a stochastic process will obfuscate the gradients for finding the optimal adversarial perturbation δ, making the outer minimization (robust training) less effective. In order to sidestep such stochasticity, we propose the *deterministic attack* that creates a deterministic gradient path within the model to generate the adversarial perturbation. $\hat{\mathbf{Z}}$. Specifically, we use a *deterministic latent code* by replacing the sampling process $\mathbf{Z}^{(k)} \sim \mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}),\, p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$, with the maximum-likelihood sample (here, i.e $\hat{\mathbf{Z}} = p_\theta^\mu(\mathbf{Z}|\mathbf{X})$). The objective for generating the adversarial perturbation is thus:

$$\delta = \arg\max_{\delta\in\mathbb{S}} \mathcal{L}_{\text{adv}}(\mathbf{X}+\delta, \mathbf{Y}) = \arg\max_{\delta\in\mathbb{S}} \parallel p_\theta(\mathbf{Y}|\hat{\mathbf{Z}}, \mathbf{X}+\delta) - \mathbf{Y} \parallel^2 , \text{ where } \hat{\mathbf{Z}} = p_\theta^\mu(\mathbf{Z}|\mathbf{X}+\delta). \tag{4}$$

We empirically show that gradients from this deterministic gradient path can effectively guide the generation of adversarial examples. We name our attack as *Deterministic Attack*.

Stabilize adversarial training with bounded noise and hybrid objective. During the adversarial training process, the distribution of the perturbed input $\mathbf{X} + \delta$ coevolves with the training process as δ is calculated via an inner maximization process at each training iteration. Although δ is bounded



(a) Clean (b) Salt and pepper (c) Adv noise (d) Correlation

Figure 2: Visualizations of the CVAE models trained with clean (a) data, *Salt and pepper* noise (b), and adversarial perturbations (c); Quantitative results of the correlation between the label of the generated images and conditioned images at different noise levels (d).

by the adversarial set $\mathbb{S}_p^\epsilon$, the resulting latent condition variable $\mathbf{C} = f(\mathbf{X} + \delta)$ can be arbitrarily noisy since the Lipschitz constant of neural network layers (f) is not bounded during training (See *Lemma 1.* in Appendix A). Since $\mathbf{C} = f(\mathbf{X} + \delta)$ is noisy, it is a less informative signal compared to the deterministic signal X. Thus, modeling $p_\theta(\mathbf{Y}|\mathbf{X} + \delta, \mathbf{Z})$ becomes substantially harder. In an extreme case that $\mathbf{C} = f(\mathbf{X} + \delta)$ is super noisy and contains no information, the training process can degenerate to model $p_\theta(\mathbf{Y}|\mathbf{Z})$, resulting in the undesirable worse performance on the clean data.

To further validate the above hypothesis that it is hard to model $p_\theta(\mathbf{Y}|\mathbf{X} + \delta, \mathbf{Z})$ with a changing data distribution of $\mathbf{X}+\delta$, we conduct an additional experiment. For simplicity, we use MNIST [25] as the dataset. As shown in Fig. 2, we divide each digit image into four quadrants. We take the top-left quadrant as the condition $\mathbf{X}$ and the remaining quadrants as the output $\mathbf{Y}$. We train a CVAE ($p_\theta(\mathbf{Y}|\mathbf{X} + \delta, \mathbf{Z})$) to model $\mathbf{Y}$ by using clean data ($\mathbf{X}$) or noisy data ($\mathbf{X} + \delta$), where δ represents salt and pepper noise [26] or adversarial noise [8], resulting in Fig. 2 (a), (b), (c) respectively. The top-left region of each image in the first row is the conditional variables $\mathbf{X}$. The rest of rows are the generated images with different $\mathbf{Z}$. Each column in the same row uses the same $\mathbf{Z}$. We observe that the model trained on clean data successfully captures the conditional distribution (i.e., the generated image highly depends on $\mathbf{X}$) while the model trained with adversarial noise degenerates and ignores the condition (i.e., each row generates images of the same digit). This result shows that the conditional generative model fails to learn from $\mathbf{X}$. To provide a quantitative analysis, we measure the correlation between the label of the generated images and the label of their conditioned image quadrants, resulting in Fig 2 (d). More details on how to calculate the correlation are in the Appendix A. We observe that the correlation drops as the noise level increases for both adversarial nose and *salt and pepper* noise. Adversarial noise is more effective to degenerate the conditional generative model. Therefore, we conclude that (1) the noises in the conditional data lead to degenerated conditional generative model (i.e., from CVAE to VAE); (2) the level of degeneration depends on the noise levels.

Based on the analysis result, to better learn a robust trajectory prediction model, we need to bound $|f(\mathbf{X} + \delta) - f(\mathbf{X})|$ to reduce the noise level. Hence, we propose the following regularization loss $\mathcal{L}_{\text{reg}}$:

$$\mathcal{L}_{\text{reg}} = \mathrm{d}(f(\mathbf{X} + \delta), f(\mathbf{X})), \tag{5}$$

where d is a distance function (e.g., we use L_2 norm as the distance metric).

In addition, because the clean data has a fixed distribution, simultaneously learning from the clean data during the adversarial training process anchors the conditional distribution on a stable clean data distribution. Specifically, we propose the following hybrid objective:

$$\mathcal{L}_{\text{clean}}(\mathbf{X}, \mathbf{Y}) = \mathcal{L}_{\text{total}}(\mathbf{X}, \mathbf{Y}), \tag{6}$$

where $\mathcal{L}_{\text{total}}$ could be the loss in Eq. 1 for CVAE-based model or Eq. 2 for cGAN-based model.

Protect benign performance using data augmentation. Adversarial training often leads to performance degradation on clean data [23]. However, trajectory prediction is a critical component for safety-critical AD systems and its performance degradation can result in severe consequences (e.g., collisions). Thus, it is important to balance the model performance on the clean and adversarial data when designing adversarial training algorithms.

To further improve the performance on clean and adversarial data, we need to address the overfitting problem of the min-max adversarial training [27]. Data augmentation is shown to be effective in addressing the problem in the image classification domain [28]. However, data augmentation in trajectory prediction is rarely studied and non-trivial. To design an effective augmentation algorithm, Rebuffi et al. [28] argues that the most important criterion is that the augmented data should be realistic and diverse. Thus, we design a dynamic-model based data augmentation strategy $\mathbb{A}$ shown in the Appendix A. By using the augmentation, we can generate diverse, realistic multi-agent trajectories for each scene and construct $\mathbb{D}_{\text{aug}}$.

RobustTraj. In summary, our adversarial training strategy for trajectory prediction models is formulated as follows:

$$\begin{aligned} \delta &= \arg\max_{\delta \in \mathbb{S}} \mathcal{L}_{\text{adv}}(\mathbf{X}+\delta, \mathbf{Y}), \quad \text{where}\{\mathbf{X}, \mathbf{Y}\} \in \mathbb{D} \cup \mathbb{D}_{\text{aug}} \\ \theta,\, \phi &= \arg\min_{\theta,\, \phi} \mathcal{L}_{\text{total}}(\mathbf{X}+\delta, \mathbf{Y}) + \mathcal{L}_{\text{clean}}(\mathbf{X}, \mathbf{Y}) + \beta \cdot \mathcal{L}_{\text{reg}}(\mathbf{X},\, \mathbf{X}+\delta), \end{aligned} \tag{7}$$

where $\mathbb{D}$, $\mathbb{D}_{\text{aug}}$ are the training data and augmented data; $\mathcal{L}_{\text{adv}}$ is adversarial loss to generate effective adversarial examples in Eq. 4; $\mathcal{L}_{\text{total}}$ is the loss in Eq. 1 or Eq. 2 to train a robust model against adversarial examples; $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{clean}}$ are loss shown in Eq. 5 and Eq. 6 to provide a stable signal for training. β is a hyper-parameter for adjusting the regularization.

5 Experiments and Results

5.1 Experimental setup

Dataset and models. We follow the setting in prior work [4, 3] and use the nuScenes dataset [29] for evaluation. For the trajectory prediction models, we select the representative conditional generative models based on CVAE (AgentFormer [4]) and cGAN (Social-GAN [1]). AgentFormer is a state-of-the-art model based on CVAE and Social-GAN is a classic model based on cGAN. We report the final results for all three models: AgentFormer (AF), mini-AgentFormer (mini-AF) and Social-GAN. More details are shown in the Appendix B.

Training details and hyperparameter choices. For the adversarial training, we choose a 2-step Projected Gradient Descent (PGD) attack for the inner maximization and choose $\beta = 0.1$. We train 50 epochs and 100 epochs for AgentFormer and Social-GAN respectively. For other hyperparameters during training, we follow the original settings for AgentFormer and Social-GAN. The details for choosing these hyperparameters can be found in the Appendix B. All experiments are done on the NVIDIA V100 GPU [30]. We consider various baselines, including naïve adversarial training (naïve AT) and four defenses proposed by Zhang *et al.* [11]: data augmentation with adversarial examples (*DA*), *train-time smoothing*, *test-time smoothing*, *DA + train-time smoothing* and *detection + test-time smoothing*.

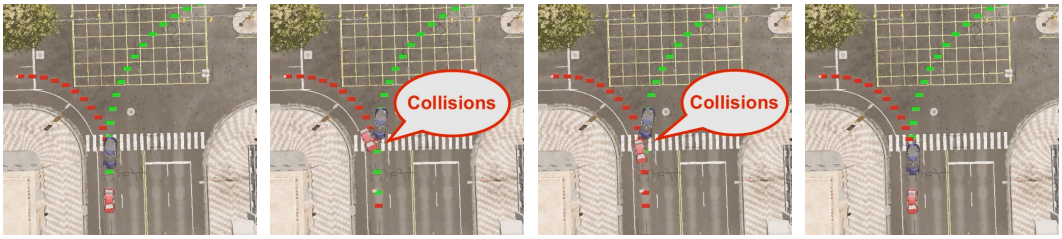
Attack and evaluation metrics. For the adversarial attack, we choose a 20-step PGD attack (an ablation study on step convergence can be found in the Appendix B). Without loss of generality, we use L_∞ as the attack threat model so that $\mathbb{S} = \{\delta | \parallel \delta \parallel_\infty \leq \epsilon\}$. We select $\epsilon = \{0.5, 1.0\}$-meter, where the 1-meter deviation is the maximum change for a standard car without shifting to another lanes [11]. We use four standard evaluation metrics for the nuscenes prediction challenge [29]: average displacement error (ADE), final displacement error (FDE), off road rates (ORR), and miss rate (MR). We evaluate the model's performance on both clean and adversarial data. For convenience, we use *ADE*, *FDE*, *ORR*, *MR* to represent the performance on the clean data and *Robust ADE*, *Robust FDE*, *Robust ORR*, *Robust MR* to represent the performance under attacks. We compute these metrics with the best of five predicted trajectory samples, i.e., $K = 5$.

5.2 Main results

Here, we present our main results of *RobustTraj*. We compare it with the baselines including model trained with clean data (*Clean*) and naïve adversarial training (*Naïve AT*), and existing defense methods for trajectory prediction [11]. The results have been shown in Table 1.

Table 1: ADE and Robust ADE on different defense methods and models. The 1-st and 2-nd lowest errors are colored.

Model	mini-AF				AF				SGAN			
Method	ADE		Robust ADE		ADE		Robust ADE		ADE		Robust ADE	
	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
Clean	**2.05**	**2.05**	6.86	11.53	**1.86**	**1.86**	5.09	8.57	**4.80**	**4.80**	10.52	20.15
Naïve AT [8]	2.75	2.78	5.44	9.20	2.52	2.56	**3.81**	6.81	6.43	6.55	**8.34**	**14.63**
DA [11]	2.31	2.32	5.54	9.32	2.10	2.08	4.35	7.22	5.41	5.40	8.85	17.25
Train-time Smoothing [11]	3.14	3.07	5.67	9.31	2.11	2.13	4.19	6.79	5.50	5.47	8.74	16.51
Test-time Smoothing [11]	2.97	3.07	**4.96**	**8.50**	2.40	2.41	4.43	7.44	6.16	6.17	9.05	17.42
DA + Train-time Smoothing [11]	2.41	2.39	5.48	9.00	2.17	2.13	4.14	**6.62**	5.63	5.61	8.60	16.14
Detection + Test Smoothing [11]	2.31	2.28	5.91	9.85	2.08	2.03	4.45	7.59	5.35	5.37	9.28	17.39
RobustTraj	**2.14**	**2.11**	**3.69**	**3.82**	**1.91**	**1.95**	**2.73**	**2.86**	**4.95**	**5.07**	**5.20**	**6.94**



(a) Benign case (b) Adv Attack (c) w/ defense [11] (d) w/ defense (ours)

Figure 3: Impacts to a MPC-basd downstream planner. (a) is under the benign case while (b), (c) and (d) are under the adversarial attacks. The blue car and the red car represent the AV and the adversarial agent respectively.

We observe that our method achieves the best robustness and maintains good clean performance for most cases. For instance, with $\epsilon = 0.5$ attack on AgentFormer model, our method is able to reduce 46% prediction errors ($\frac{5.09-2.73}{5.09}$) under the attack at a cost of 2.6% ($\frac{1.91-1.86}{1.86}$) clean performance degradation on ADE, compared to the model trained with clean data at $\epsilon = 0.5$. Compared to the existing methods, our method also significantly outperforms in terms of the robustness. For instance, with $\epsilon = 1.0$ attack on AgentFormer model, our method achieves 45% better robustness with 9% better clean performance on ADE compared to the best results from existing methods [11].

Impacts to downstream planners. To further study the downstream impact of our robust trajectory model in the AD stack, we plug it into a planner. We select a MPC-based planner and evaluate the collision rates under the attack. To perform the attack on a closed-loop planner, we conduct attacks on a sequence of frames with the expectation over transformation (EOT) [31] method. We follow the setting from Zhang et al. [11] and choose $\epsilon = 1$. We choose AgentFormer model since it has the most competitive performance. As a result, we observe that, while AgentFormer model trained on clean data leads to 10 collision cases under attack, the robust trained model with the proposed *RobustTraj* is able to avoid all the collisions. As shown in Fig. 3, we demonstrate that the proposed *RobustTraj* is able to avoid the collisions (Fig. 3 (d)) while the *DA + Train-time Smoothing* method proposed by Zhang et al. [11] is not (Fig. 3 (c)).

5.3 Component analysis

In this section, we analyze the effectiveness of the three components. We use the mini-AgentFormer model since it has competitive performance and is lightweight for a fast adversarial training process.

Effectiveness of the *Deterministic Attack*. To demonstrate the importance of the *Deterministic Attack*, we compare it with competitive alternatives, *Latent Attack* and *Context Attack*, which also construct the deterministic path. However, they only attack a partial model as opposed to our end-to-end full model attack. More details about these attacks are in the Appendix A. We evaluate their attack effectiveness by attacking a normally trained trajectory prediction model (without robust training).

In Fig. 4, we demonstrate that *Deterministic Attack* is the most effective attack among all. Additionally, we embed them into the whole adversarial training pipeline and evaluate the adversarial robustness. The results are shown in Table 2. We observe that the model trained with the *Deterministic Attack* achieves the best robustness in terms of ADE. More results with other metrics and the other ϵ are in the Appendix B.

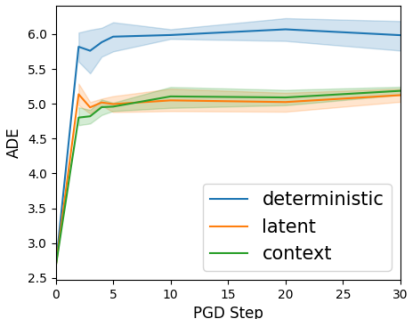


Figure 4: Peformance of different attacks in mini-AgenetFormer.

Effect of additional loss functions. We evaluate the performance of the models trained with additional loss terms: $\mathcal{L}_{\text{clean}}$ and $\mathcal{L}_{\text{reg}}$. In Table 2, we can see that the regularization term $\mathcal{L}_{\text{reg}}$ improves robustness of the models and achieves better clean performance. It shows that the regularization of the introduced noises on conditional variables help the model to stabilize the training procedure. By adding the clean loss $\mathcal{L}_{\text{clean}}$, we observe that both the robustness and clean performance are improved further, which means the benign data indeed anchors the model output on clean data distribution and provides a stable signal for the better robust training for generative models.

Effect of domain-specific augmentation. To demonstrate the effectiveness of the domain-specific augmentation, We also combine it with all of the above components to validate its effect. The results are shown in Table 2. We observe that it achieves a better performance on clean and adversarial data.

Table 2: ADE and robust ADE for different methods on mini-AgentFormer. The lowest error is in bold.

Method	ADE		Robust ADE	
	0.5	1.0	0.5	1.0
Clean	2.05	2.05	6.86	11.53
Latent Attack	2.55	2.70	4.10	4.71
Context Attack	**2.47**	2.59	3.94	4.78
Deterministic Attack	2.61	**2.55**	**3.88**	**4.35**
Deterministic Attack				
+ $\mathcal{L}_{\text{reg}}$	2.29	2.31	3.76	4.28
+ $\mathcal{L}_{\text{clean}}$ + $\mathcal{L}_{\text{reg}}$	2.23	2.19	3.71	3.83
+ $\mathcal{L}_{\text{clean}}$ + $\mathcal{L}_{\text{reg}}$ + Aug	**2.14**	**2.11**	**3.69**	**3.82**

6 Limitations

In this work, we identified the challenges of applying adversarial training on trajectory prediction models based on conditional generative models. Though the conditional generative model is the main-stream architecture for the trajectory prediction task, there are other architectures (e.g., flow-based method [5, 6] and RL-based method [32]) for generating multi-modal predictions. We leave these as future works for building robust trajectory prediction models.

7 Conclusion

In this paper, we aim to study how to train robust generative trajectory prediction models against adversarial attacks, which is seldom explored in the literature. To achieve this goal, we first identify three key challenges in designing an adversarial training framework to train robust trajectory prediction models. To address them, we propose an adversarial training framework with three main components, including (1) a *deterministic attack* for the inner maximization process of the adversarial training, (2) additional regularization terms for stable outer minimization of adversarial training, and (3) a domain-specific augmentation strategy to achieve a better performance trade-off on clean and adversarial data. To show the generality of our method, we apply our approach to two trajectory prediction models, including (1) a CVAE-based model, AgentFormer, and (2) a cGAN-based model, Social-GAN. Our extensive experiments show our method could significantly improve the robustness with a slight performance degradation on the clean data, compared to the existing techniques and dramatically reduce the severe collision rates when plugged into the AD stack with a planner. We hope our work can shed light on developing robust trajectory prediction systems for AD.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[2] B. Ivanovic and M. Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019.

[3] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

[4] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[5] N. Rhinehart, K. M. Kitani, and P. Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

[6] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2821–2830, 2019.

[7] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[10] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[11] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. *arXiv preprint arXiv:2201.05057*, 2022.

[12] A. Jeddi, M. J. Shafiee, and A. Wong. A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning. *arXiv preprint arXiv:2012.13628*, 2020.

[13] D. Liu, R. Yu, and H. Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283. IEEE, 2019.

[14] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[15] N. Papernot and P. McDaniel. Extending defensive distillation. *arXiv preprint arXiv:1705.05264*, 2017.

[16] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

[17] E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[18] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[19] C. Xie and A. Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HyxJhCEFDS.

[20] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

[21] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[22] Y. Yang, G. Zhang, D. Katabi, and Z. Xu. Me-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019.

[23] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[24] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283. PMLR, 2018.

[25] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[26] Wikipedia contributors. Salt-and-pepper noise — Wikipedia, the free encyclopedia, 2022. URL https://en.wikipedia.org/w/index.php?title=Salt-and-pepper_noise&oldid=1085283109. [Online; accessed 16-June-2022].

[27] L. Rice, E. Wong, and Z. Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

[28] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

[29] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[30] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021. doi:10.1109/MM.2021.3061394.

[31] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[32] N. Deo and M. M. Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020.