Spring 5-16-2023

Targeted Adversarial Attacks against Neural Network Trajectory Predictors

Kaiyuan Tan

WASHINGTON UNIVERSITY IN ST. LOUIS

McKelvey School of Engineering
Department of Electrical & Systems Engineering

Thesis Examination Committee:
Yiannis Kantaros, Chair
ShiNung Ching
Shen Zeng

Targeted Adversarial Attacks against Neural Network Trajectory Predictors

by
Kaiyuan Tan

A thesis presented to
the McKelvey School of Engineering
of Washington University in
partial fulfillment of the
requirements for the degree
of Master of Science

May 2023
St. Louis, Missouri

Table of Contents

List of Figures . . . . . **iii**

List of Tables . . . . . **iv**

Acknowledgments . . . . . **v**

Abstract . . . . . **vi**

Chapter 1: Introduction **1**

Chapter 2: Problem Formulation **4**
2.0.1 Trajectory Prediction via Deep Neural Networks 4
2.0.2 Targeted Adversarial Attack Formulation 5

Chapter 3: Proposed Targeted Adversarial Attack for Trajectory Prediction **7**

Chapter 4: Experiments **9**
4.0.1 Experimental Setup 9
4.0.2 Evaluation of TA4TP 11

Chapter 5: Some future work attempts **16**

Chapter 6: Conclusion **17**

References **18**

List of Figures

Figure 1.1: A graphical demonstration of the proposed *targeted* adversarial attack in an autonomous driving (left) and pursuit evasion scenario (right). In the left figure, the adversary (car A) designs a trajectory so that the DNN model of car C predicts that car A will accelerate and move between cars B and C. The latter may cause the nearby moving cars make unsafe decisions (e.g., accelerate or even exit their road lanes). In the right figure, the red marked drone plans to move towards a restricted area to take pictures of factory facilities. The green marked drone collects past trajectories of nearby moving agents and, using a DNN model, predicts their future paths. If the predicted trajectories lead towards the restricted area, an alarm is raised, and the green drone is tasked with pursuing the intruders. One of the strategies that the red drone applies to remain stealthy is to follow adversarially crafted trajectories that will make the green drone specifically predict that the red drone is heading away from the restricted area. 2

Figure 2.1: Graphical illustration of the problem formulation for $P = F = 4$. Our goal is to design an adversarial input trajectory (red solid line) that looks natural (i.e., close to nominal inputs - blue solid line) and fools the DNN model into predicting a trajectory (red dashed line) that is as close as possible to a desired/target trajectory (cyan dashed line). 5

Figure 4.1: Graphical illustration of TA4TP in three different scenarios. Observe that in all cases the perturbed input (red solid line) is very close to the nominal trajectory (blue solid line) while the predicted trajectory (red dashed) almost overlaps with the target one (cyan dashed). 11

Figure 4.2: Performance of TA4TP given an aggressive target trajectory. The target trajectory requires the car to move along the cyan direction with a velocity that is significantly higher than the one associated with the input path. This difference in the velocity is illustrated by the large distance between the waypoints in the target trajectory. . . 12

List of Tables

Table 4.1: Summary of results for TA4TP with $K_{\max} = 100$ and $\tau = 0.02$m. The second, third, and fourth column show the nominal accuracy of the DNN models, the average deviation between the target and the ground truth trajectories, and the average deviation between the predicted and the target trajectories, respectively. The last two columns show the average runtime to design a single adversarial trajectory. 10

Table 4.2: Summary of results for TA4TP with $K_{\max} = 10$ and $\tau = 0.02$m. 13

Table 4.3: DNN robustness analysis against random noise on clean inputs. 13

Acknowledgments

The whole project was finished on a tight schedule. It was a challenge for me both physically and mentally. I was eager to prove myself before the application season, so this project was quite meaningful from my perspective. I will always treasure those late nights I've been working. Not only because that work sharpens my coding skills, but it also reminds me of the countless help my advisor and friend had given.

First and foremost, I want to say thank you to my dear advisor. He offered professional advice as well as instructions during the whole project, making it an innovative work. Although I might be the early-stage student working with him, he showed his patience and in-depth vision during the process. I believe he would be a great professor in the future. Secondly, I'm willing to acknowledge my senior Wang in this part. When he knew I didn't have a place to do research before my Ph.D. application, he referred me to his group where we had lots of cooperation later. To some extent, this paper might not exist without him. Last but not least, I would like to appreciate myself. Facing a bunch of codes was challenging, but I made it.

At the time when I was writing this paragraph, I had already declined two offers this year. And decided to serve as a RA for one year. This might be a tough decision that leads to nowhere, but I made up my mind to give my dream a try. Hope I can get my dream offer in the next year!

Kaiyuan Tan

Washington University in St. Louis
May 2023

ABSTRACT OF THE THESIS

Targeted Adversarial Attacks against Neural Network Trajectory Predictors

by

Kaiyuan Tan

Master of Science in Electrical Engineering

Washington University in St. Louis, 2023

Assistant Professor Yiannis Kantaros, Chair

Trajectory prediction is an integral component of modern autonomous systems as it allows for envisioning future intentions of nearby moving agents. Due to the lack of other agents' dynamics and control policies, deep neural network (DNN) models are often employed for trajectory forecasting tasks. Although there exists an extensive literature on improving the accuracy of these models, there is a very limited number of works studying their robustness against adversarially crafted input trajectories. To bridge this gap, in this paper, we propose a targeted adversarial attack against DNN models for trajectory forecasting tasks. We call the proposed attack TA4TP for Targeted adversarial Attack for Trajectory Prediction. Our approach generates adversarial input trajectories that are capable of fooling DNN models into predicting user-specified target/desired trajectories. Our attack relies on solving a nonlinear constrained optimization problem where the objective function captures the deviation of the predicted trajectory from a target one while the constraints model physical requirements that the adversarial input should satisfy. The latter ensures that the inputs look natural and they are safe to execute (e.g., they are close to nominal inputs and away from obstacles). We demonstrate the effectiveness of TA4TP on two state-of-the-art DNN models and two datasets. To the best of our knowledge, we propose the first targeted adversarial attack against DNN models used for trajectory forecasting.

Chapter 1

Introduction

Trajectory prediction algorithms play a pivotal role in enabling autonomous systems to make safe and efficient control decisions in highly dynamic environments as they can forecast future behaviors of nearby moving agents [9,11,12,15,16,21,29,33,36,38,43,45]. To address the lack of knowledge of other agents' intentions and control policies, deep neural network (DNN) models are often employed to address behavior forecasting tasks as e.g., in [6, 28, 35, 41]. These works typically assess the performance of the proposed DNN models by measuring the deviation of the predicted trajectories from the ground truth ones. However, they neglect to evaluate their robustness against adversarially crafted input trajectories. In fact, lack of adversarial robustness can significantly compromise safety of autonomous systems; see e.g., the autonomous driving and pursuit-evasion scenarios shown in Fig. 1.1.

A first step towards evaluating robustness of trajectory prediction models is to provide automated methods that compute adversarial inputs (i.e., corner cases in the input space) where these models fail. To this end, in this paper, we propose a new white box targeted adversarial attack against DNN models used for trajectory forecasting tasks. We call the proposed attack TA4TP for Targeted adversarial Attack for Trajectory Prediction. The goal of TA4TP is to perturb any nominal input trajectory so that the DNN prediction is as close as possible to a user-specified target/desired trajectory. Throughout the paper, trajectories are defined as finite sequences of system states (e.g., positions of a car). We formulate the attack design process as a constrained non-linear optimization problem where the objective function captures the deviation of the predicted trajectory from the desired one and the constraints capture physical requirements that the perturbed input should satisfy. Specifically, to define the objective function, we first assign weights to each state in the target trajectory; the higher the weight is, the more important the corresponding desired state is. This allows an adversary to assign priorities to the desired states. For instance, in certain applications it may be significant for an adversary to make other agents wrongly

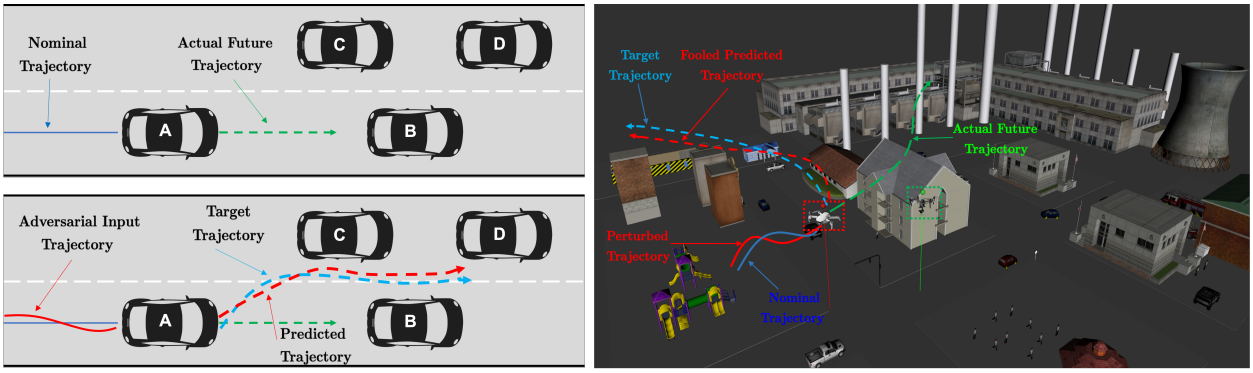

Figure 1.1: A graphical demonstration of the proposed *targeted* adversarial attack in an autonomous driving (left) and pursuit evasion scenario (right). In the left figure, the adversary (car A) designs a trajectory so that the DNN model of car C predicts that car A will accelerate and move between cars B and C. The latter may cause the nearby moving cars make unsafe decisions (e.g., accelerate or even exit their road lanes). In the right figure, the red marked drone plans to move towards a restricted area to take pictures of factory facilities. The green marked drone collects past trajectories of nearby moving agents and, using a DNN model, predicts their future paths. If the predicted trajectories lead towards the restricted area, an alarm is raised, and the green drone is tasked with pursuing the intruders. One of the strategies that the red drone applies to remain stealthy is to follow adversarially crafted trajectories that will make the green drone specifically predict that the red drone is heading away from the restricted area.

predict that its final state is within a certain region while the predicted trajectory towards that region may be of secondary importance; this is the case e.g., in the autonomous driving example shown in Fig. 1.1 and in the experiments provided in Section **??**. Then, the objective function is defined as the weighted average ℓ_2 distance between the predicted and the desired states. The constraints require the adversarially perturbed trajectory to satisfy certain physical constraints. For instance, in Fig. 1.1, in the autonomous driving scenario, the perturbed trajectory should stay within the lane and close enough to the nominal trajectory. Similarly, in the pursuit-evasion scenario shown in Fig. 1.1, the perturbed trajectory should be obstacle-free. Assuming that the structure of the target DNN model is fully known, we solve this optimization problem by leveraging gradient-based methods, such as the Adam optimizer [26]. Our experiments on state-of-the-art datasets and DNN models show that the proposed attack can successfully force given (and known to the attacker) DNN models to predict desired trajectories. We believe that the proposed attack will enable users to evaluate as well as enhance the adversarial robustness of DNN-based trajectory forecasters.

Related Works: DNNs have seen renewed interest in the last decade due to the vast amount of available data and recent advances in computing. In autonomous systems, DNNs are typically used either as feedback controllers and planners [1,8,13,39], perception modules [31,40], or for trajectory prediction [6,28,41] that is also the case in this paper. Despite the impressive experimental performance of DNNs, their brittleness has resulted in unreliable system behaviors and public failures preventing their wide adoption in safety-critical applications. This is also demonstrated by several adversarial attack algorithms that have been proposed recently. These attacks, similar to the proposed one, aim to minimally manipulate inputs to DNN models, so that they can cause incorrect outputs that would benefit an adversary. The large majority of existing adversarial attacks against DNN models are focused on perceptual tasks such as image classification or object detection as e.g., in [2,5,7,10,14,27,32,37]. Recently, adversarial attacks against DNNs used for planning and control have been proposed in [17,19,42]. However, there is a very limited number of studies evaluating robustness of DNN models for trajectory prediction against adversarial attacks. We believe that the closest works to ours are the recent ones presented in [4,44]. Common in these works is that they design *untargeted* attacks, i.e., they aim to maximize the prediction error or, in other words, the difference between predicted and ground truth trajectories. To the contrary, in this work we design *targeted* adversarial attacks to make DNN predictions be as close as possible to *any* user-specified desired trajectories. We argue that the proposed targeted attack is more expressive than un-targeted ones as the latter do not allow the adversary to freely pick any desired predicted trajectory and, therefore, cause desired unsafe situations. For instance, using untargeted attacks, in the autonomous driving setup in Fig. 1.1, an adversarially crafted trajectory for car A that maximizes the prediction error may point to the left or right lane which may not necessarily compromise safety of other cars. To the contrary, the proposed attack allows the adversary to select target trajectories, as shown in Fig. 1.1, that may force other cars make unsafe decisions. To the best of our knowledge, we propose the first targeted adversarial attack against trajectory forecasting DNN models.

Chapter 2

Problem Formulation

In this section, we first describe the trajectory prediction task (Section 2.0.1) and then we formally define the targeted adversarial attack design problem as a nonlinear constrained optimization problem (Section 2.0.2).

2.0.1 Trajectory Prediction via Deep Neural Networks

We consider trajectory prediction tasks accomplished by DNNs. The goal in these tasks is to forecast the future trajectory of an agent given its past trajectories. Particularly, a DNN model takes as an input a sequence of P past observed states of a moving agent (e.g., locations of a pedestrian) every T time units, and outputs a sequence of predicted future states of this agent; see Fig. 2.1. We denote the input sequence to the DNN model by $\mathbf{X}_{t-P:t} = [\mathbf{X}_{t-P}, ..., \mathbf{X}_{t-1}, \mathbf{X}_t]$, where $\mathbf{X}_n$ is the state of the agent at the past time step $n \in [t-P, \ldots, t]$. We also denote the ground truth future path of this agent in the next F future steps as $\mathbf{G}_{t+1:t+F} = [\mathbf{G}_{t+1}, \mathbf{G}_{t+2}, ..., \mathbf{G}_{t+F}]$, where $\mathbf{G}_n$ stands for the ground truth state at the future time $n \in [t+1, \ldots, t+F]$. Similarly, we denote the prediction of the DNN model for next F steps by $\mathbf{P}_{t+1:t+F} = [\mathbf{P}_{t+1}, \mathbf{P}_{t+2}, ..., \mathbf{P}_{t+F}]$. Denoting the DNN model by f, we have that $\mathbf{P}_{t+1:t+F} = f(\mathbf{X}_{t-P:t})$.

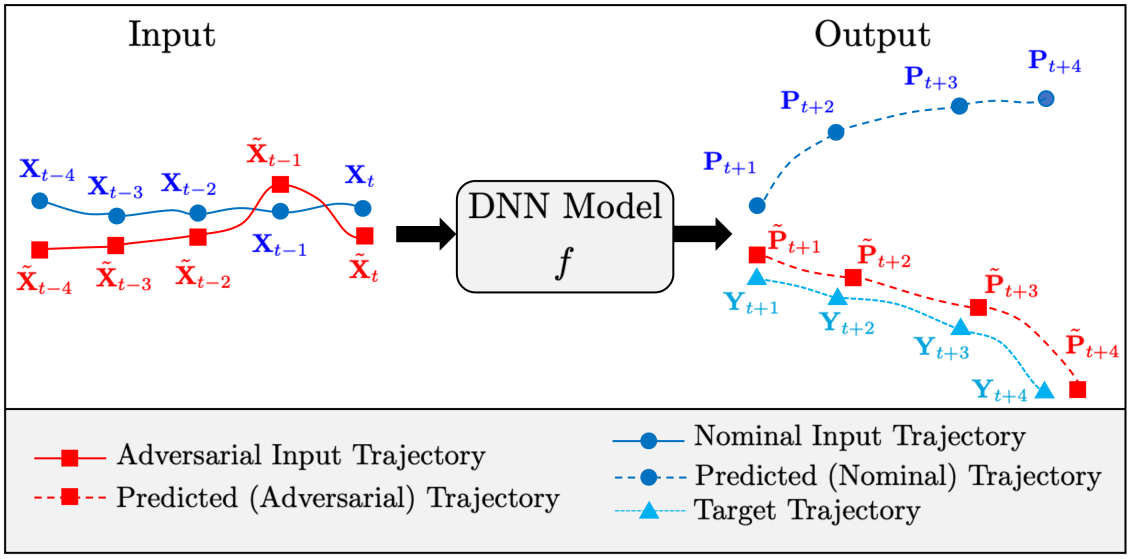

Figure 2.1: Graphical illustration of the problem formulation for $P = F = 4$. Our goal is to design an adversarial input trajectory (red solid line) that looks natural (i.e., close to nominal inputs - blue solid line) and fools the DNN model into predicting a trajectory (red dashed line) that is as close as possible to a desired/target trajectory (cyan dashed line).

2.0.2 Targeted Adversarial Attack Formulation

Consider a DNN trajectory prediction model f and any nominal trajectory $\mathbf{X}_{t-P:t}$ that the system has designed to follow in the time interval $[t-P, t]$. We note that $\mathbf{X}_{t-P:t}$ can be designed using any existing planning algorithm such as [20, 23, 24]. Our goal is to design a perturbation $\mathbf{\Delta}_{t-P:t} = [\mathbf{\Delta}_{t-P}, \ldots, \mathbf{\Delta}_{t-1}, \mathbf{\Delta}_t]$, yielding a perturbed/adversarial trajectory $\tilde{\mathbf{X}}_{t-P:t} = [\tilde{\mathbf{X}}_{t-P}, \ldots, \tilde{\mathbf{X}}_{t-1}, \tilde{\mathbf{X}}_t]$, defined as $\tilde{\mathbf{X}}_{t-P:t} = \mathbf{X}_{t-P:t} + \mathbf{\Delta}_{t-P:t} = [\mathbf{X}_{t-P} + \mathbf{\Delta}_{t-P}, ..., \mathbf{X}_{t-1} + \mathbf{\Delta}_{t-1}, \mathbf{X}_t + \mathbf{\Delta}_t]$, so that (i) if the adversarial agent follows the trajectory $\tilde{\mathbf{X}}_{t-P:t}$, then the corresponding trajectory predicted by f, denoted by $\tilde{\mathbf{P}}_{t+1:t+F} = f(\tilde{\mathbf{X}}_{t-P:t})$, will be the desired/target trajectory denoted by $\mathbf{Y}_{t+1:t+F} = [\mathbf{Y}_{t+1}\mathbf{Y}_{t+2}, ...\mathbf{Y}_{t+F}]$ (that may be completely different from the ground truth one), i.e., $\tilde{\mathbf{P}}_{t+1:t+F} = \mathbf{Y}_{t+1:t+F}$ and (ii) $\tilde{\mathbf{X}}_{t-P:t}$ satisfies certain physical constraints; see Fig. 2.1. To design this attack (i.e., $\tilde{\mathbf{X}}_{t-P:t}$), we assume that the attacker has full knowledge of the DNN model f. Formally, we formulate the targeted adversarial attack design problem as a nonlinear optimization problem defined

as follows:

$$\min_{\boldsymbol{\Delta}_{t-P:t}} J(\boldsymbol{\Delta}_{t-P:t}) = \sum_{m=t+1}^{t+F} w_m \|\tilde{\mathbf{P}}_m - \mathbf{Y}_m\|_2 \tag{2.1a}$$

$$\tilde{\mathbf{X}}_n \in \mathcal{C}_n, \forall n \in [t-P, \ldots, t] \tag{2.1b}$$

where, $\tilde{\mathbf{P}}_{t+1:t+F} = f(\tilde{\mathbf{X}}_{t-P:t}) = f(\mathbf{X}_{t-P:t} + \boldsymbol{\Delta}_{t-P:t})$. The objective function in (2.1a) captures the weighted average distance, using the ℓ_2 norm, between the predicted trajectory (i.e., $\tilde{\mathbf{P}}_{t+1:t+F}$) and the desired trajectory (i.e., $\mathbf{Y}_{t+1:t+F}$). Also, in (2.1a), w_m is a weight modeling the importance of the m-th state (i.e., $\mathbf{Y}_m$) in the desired trajectory $\mathbf{Y}_{t+1:t+F}$; the higher the w_m is, the more important is for $\tilde{\mathbf{P}}_m$ to be close to $\mathbf{Y}_m$. The weights are selected so that $w_m \in [0, 1]$ and $\sum_{m=t+1}^{t+F} w_m = 1$. The constraint $\tilde{\mathbf{X}}_n \in \mathcal{C}_n$, for all $n \in [t-P, \ldots, t]$, requires each state $\tilde{\mathbf{X}}_n$ to belong to a set $\mathcal{C}_n$ collecting all permissible values. For instance, in an autonomous driving scenario, if $\mathbf{X}_n$ captures the agent position, then $\tilde{\mathbf{X}}_n \in \mathcal{C}_n$ may require the adversarially crafted trajectory to be fully within the lane (to ensure safety of the adversary). Note that, in general, the sets $\mathcal{C}_n$ can be defined differently across the states of the trajectories while their design is scenario-specific. We assume that all states in the nominal trajectory satisfy the corresponding constraints and, therefore, (2.1) is feasible; e.g., zero perturbation is a feasible solution. In summary, in this paper, we address the following problem: *Given* (i) a fully known trajectory prediction DNN model f; (ii) a nominal trajectory $\mathbf{X}_{t-P:t}$ that the system will follow in the time interval $[t-P:t]$; (iii) a desired predicted trajectory $\mathbf{Y}_{t+1:t+F}$; (iv) weights w_m for all $m \in [t+1, \ldots, t+F]$ and a set of permissible states $\mathcal{C}_n$ for all $n \in [t-P, \ldots, t]$, *compute* the perturbation $\boldsymbol{\Delta}_{t-P:t}$ that once applied to $\mathbf{X}_{t-P:t}$ it will minimize the average weighted deviation between the DNN prediction (i.e., $\tilde{\mathbf{P}}_{t+1:t+F} = f(\mathbf{X}_{t-P:t} + \boldsymbol{\Delta}_{t-P:t})$) and the desired trajectory (i.e., $\mathbf{Y}_{t+1:t+F}$), as captured by (2.1).

Chapter 3

Proposed Targeted Adversarial Attack for Trajectory Prediction

In this section, we present our approach to address Problem 2.0.2. In the rest of this section, for simplicity of notation, when it is clear from the context, we drop the dependence of trajectories on time. For instance, we simply denote the nominal trajectory by $\mathbf{X}$ instead of $\mathbf{X}_{t-P:t}$. This extends to all sequences of states and perturbation (e.g., $\mathbf{\Delta}$, $\tilde{\mathbf{X}}$, $\mathbf{P}$, $\mathbf{Y}$).

The proposed adversarial attack, called TA4TP, leverages iterative gradient-based algorithms; see Algorithm 1. We denote by $\mathbf{\Delta}^k$ the perturbation generated by Algorithm 1 at iteration k. First, we randomly initialize the perturbation, denoted by $\mathbf{\Delta}^0$. Then, at every iteration k of the algorithm we update $\mathbf{\Delta}^k$ by moving along a descent direction that minimizes the loss function $J(\mathbf{\Delta})$. This can be achieved by simply applying a gradient descent step i.e.,

$$\mathbf{\Delta}^{k+1} = \mathbf{\Delta}^k - \epsilon_k \nabla J(\mathbf{\Delta}^k), \tag{3.1}$$

where ϵ_k is a step-size. We note that any other optimization algorithm can be used to compute $\mathbf{\Delta}^k$ so that $J(\mathbf{\Delta}_{k+1}) \leq J(\mathbf{\Delta}_k)$, such as the Adam optimizer; see Section **??**. Then, we compute the corresponding perturbed trajectory as $\tilde{\mathbf{X}}^{k+1} = \mathbf{X} + \mathbf{\Delta}^{k+1}$.

Next, we check if all states in $\tilde{\mathbf{X}}^{k+1}$ satisfy the constraints captured in (2.1b), i.e., if $\tilde{\mathbf{X}}_n^{k+1} \in \mathcal{C}_n$, for all n. If so, then the iteration index k is updated, i.e., $k = k + 1$ and we repeat the above process. Otherwise, we project $\tilde{\mathbf{X}}^{k+1}$ into the feasible space captured by the sets $\mathcal{C}_n$. Note that $\mathcal{C}_n$ may be high-dimensional and non-convex sets making the projection process challenging. Inspired by [44], to address this issue, we apply a simple line search algorithm. Particularly, first, we introduce parameters $\theta_n \in [0, 1]$, associated with each state $\tilde{\mathbf{X}}_n^{k+1}$. Then, we aim to find the maximum values of θ_n, so that $\tilde{\mathbf{X}}_n^{k+1} = \mathbf{X}_n + \theta_n \mathbf{\Delta}_n^{k+1}$ belongs to the

Algorithm 1 TA4TP: Targeted adversarial Attack for Trajectory Prediction

Input: {Nominal trajectory $\mathbf{X}$, DNN f, Target trajectory $\mathbf{Y}$, Physical Constraints $\mathcal{C}_n$}
Output: {Perturbed Trajectory $\tilde{\mathbf{X}}$}
Initialize ϵ_0 and $\boldsymbol{\Delta}^0$, and set $k = 0$
while ($k <= K_{\max}$) OR ($J(\boldsymbol{\Delta}^k) \leq \tau$) **do**
  Update $\boldsymbol{\Delta}^{k+1} = \boldsymbol{\Delta}^k - \epsilon_k \nabla J(\boldsymbol{\Delta}^k)$
  if $\mathbf{X} + \boldsymbol{\Delta}^{k+1}$ does not satisfy the constraints $\mathcal{C}_n$ **then**
    Compute $\boldsymbol{\theta}$ as per (3.2) (Projection)
    Compute $\boldsymbol{\Delta}^{k+1} = \boldsymbol{\theta} \circ (\boldsymbol{\Delta}^k - \epsilon_k \nabla J(\boldsymbol{\Delta}^k))$
  end if
  Current perturbed trajectory $\tilde{\mathbf{X}}^{k+1} = \mathbf{X} + \boldsymbol{\Delta}^{k+1}$
  $k = k + 1$
  Update ϵ_k
end while
Output: $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}^{k+1}$

set $\mathcal{C}_n$. In math, to perform this projection, we solve the following optimization problem:

$$\max_{\theta_{t-P},\ldots,\theta_t} \sum_{n=t-P}^{t} \theta_n \tag{3.2a}$$

$$\mathbf{X}_n + \theta_n \boldsymbol{\Delta}_n^{k+1} \in \mathcal{C}_n, \forall n \in \{t-P, \ldots, t\}, \tag{3.2b}$$

$$\theta_n \in [0, 1], \forall n \in \{t-P, \ldots, t\}. \tag{3.2c}$$

We solve (3.2) by simply applying a line search algorithm. Observe that since the nominal trajectory $\mathbf{X}$ satisfies the constraint (2.1b), we have that (3.2) is always feasible (i.e., $\theta_n = 0$ is always a feasible solution). We note that the above projection process may be sub-optimal if the sets $\mathcal{C}_n$ are non-convex in the sense that there may be other points on the boundary of $\mathcal{C}_n$ that are closer to $\mathbf{X}_n + \boldsymbol{\Delta}_n^{k+1}$ than the ones generated by solving (3.2). Once $\boldsymbol{\theta} = [\theta_{t-P}, \ldots, \theta_{t-1}, \theta_t]$ is computed, we update the perturbation as

$$\boldsymbol{\Delta}^{k+1} = \boldsymbol{\theta} \circ (\boldsymbol{\Delta}^k - \epsilon_k \nabla J(\boldsymbol{\Delta}^k)), \tag{3.3}$$

where $\circ$ denotes the Hadamard product (i.e., the element wise product between two vectors). Next, the iteration index k is updated, i.e., $k = k + 1$, and the above iteration is repeated. The algorithm terminates either after a user-specified maximum number $K_{\max}$ of iterations has been reached or when the loss function $J(\boldsymbol{\Delta})$ is below a desired threshold τ.

Chapter 4

Experiments

In this section, we evaluate the efficiency of proposed attack. In particular, in Section 4.0.1, we present the considered datasets and DNN models. In Section 4.0.2, we evaluate the performance of the designed attack under various settings.

4.0.1 Experimental Setup

Models: We consider two state-of-the-art and open-source trajectory prediction models. The first one is Grip++, proposed in [28], which achieves good performance over several datasets. Grip++ uses a graph to represent the interactions of close objects and uses an encoder-decoder long short-term memory (LSTM) model to make predictions. The second model is Trajectron++ [41], a modular, graph-structured model that predicts the trajectories of diverse agents while incorporating agent dynamics and heterogeneous data (e.g., semantic maps). The latter predicts multiple trajectories with probabilities and we select the trajectory with the highest probability as the final result.

Datasets: In our implementation, we considered two datasets: Nuscenes [3] and Apolloscape [18]. They both collect trajectories from autonomous driving scenarios in urban areas. Particularly, Nuscenes includes past trajectories with four states (i.e., $P = 4$ in Section 2.0.1), future trajectories with twelve states (i.e., $F = 12$ in Section 2.0.1), and semantic maps. Apolloscape includes past trajectories that have six states (i.e., $P = 6$) and future trajectories that also have six states (i.e., $F = 6$). To provide a fair comparison across datasets and models, we neglect the semantic maps in the Nuscenes dataset.

Physical Constraints: We require the adversarially crafted input trajectory $\tilde{\mathbf{X}}$ to satisfy a set of physical constraints. Specifically, recall that each state $\tilde{\mathbf{X}}_n$ in $\tilde{\mathbf{X}}$ must belong to a set $\mathcal{C}_n$. Given the autonomous driving nature of the conducted experiments, we design the sets $\mathcal{C}_n$

	$\bar{J}_{\text{acc}}^{\text{nom}}$(m)	$\bar{J}_{G-Y}$(m)	$\bar{J}$ (m) [Adam]	T (secs) [grad]	T (secs) [Adam]
Grip_apolloscape	0.013	2.363	0.140	33.342	19.366
Grip_nuscenes	0.246	1.527	0.111	44.173	12.553
Trajectron_apolloscape	0.152	2.698	0.284	429.333	161.728
Trajectron_nuscenes	0.450	0.937	0.031	582.667	144.648

Table 4.1: Summary of results for TA4TP with $K_{\max} = 100$ and $\tau = 0.02$m. The second, third, and fourth column show the nominal accuracy of the DNN models, the average deviation between the target and the ground truth trajectories, and the average deviation between the predicted and the target trajectories, respectively. The last two columns show the average runtime to design a single adversarial trajectory.

so that they impose constraints on the position, velocity, and acceleration (all these features are included in $\tilde{\mathbf{X}}_n$) of the adversarial vehicle, as in [44]. Specifically, first we require the perturbed positions in $\tilde{\mathbf{X}}_n$ to be within 1m from the corresponding nominal/normal positions in $\mathbf{X}_n$ for all n. Given that the urban lane width is 3.7m and the average width of cars is about 1.7m, this constraint requires a car not shifting to another lane if it is normally driving in the center of the lane. Additionally, we traverse all trajectories in the testing dataset to calculate the mean μ and standard deviation σ of (1) scalar velocity, (2) longitudinal/lateral acceleration, and (3) derivative of longitudinal/lateral acceleration. For each μ and σ, we also require the respective values of the perturbed trajectories not exceeding $\mu \pm 3\sigma$. These physical constraints essentially preclude careless driving of the adversarial agent and, as a result, they have the potential to preserve stealthiness of the attack. Finally, we specify the target trajectory $\mathbf{Y}$ by determining the desired positions for the adversarial vehicle. The remaining features in the target states in $\mathbf{Y}$ (e.g., velocity and acceleration) can be computed using the desired target positions.

Weight Assignment: As mentioned in (2.1a), weights w_m need to be assigned to each state $\mathbf{Y}_m$ in the target trajectory capturing how important is the predicted states to match with the target ones. In our setup, we define the weights so that $0 < w_m < w_{m+1}$ for all $m \in \{t+1, \ldots, t+F-1\}$ to give more importance to the final states in the desired trajectory. Specifically, we define the weights so that the loss function in (2.1a) captures the exponential moving average deviation between the predicted and the desired states. We emphasize that any other definition of weights is possible.

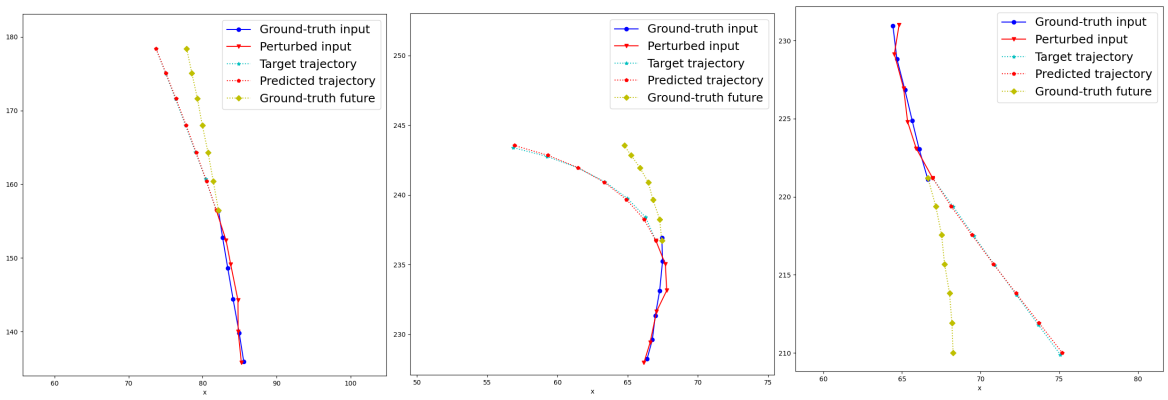

Figure 4.1: Graphical illustration of TA4TP in three different scenarios. Observe that in all cases the perturbed input (red solid line) is very close to the nominal trajectory (blue solid line) while the predicted trajectory (red dashed) almost overlaps with the target one (cyan dashed).

4.0.2 Evaluation of TA4TP

In what follows, we evaluate the performance of TA4TP on the previously described datasets and models. We randomly sample 100 scenarios as test cases from each dataset. These trajectories are called, hereafter, test trajectories.

DNN Nominal Accuracy: First, we report the performance of the DNN models in the nominal setting (i.e., without any attacks) by computing the average deviation of the predicted trajectory from the ground truth one, for every test trajectory. Specifically, we compute $J_{\text{acc}}^{\text{nom}}(\mathbf{G}, \mathbf{P}) = \sum_{m=t+1}^{t+F} \frac{\|\mathbf{P}_m - \mathbf{G}_m\|_2}{F}$ for each input test trajectory $\mathbf{X}$, where recall that $\mathbf{P}$ is the DNN prediction given the input $\mathbf{X}$. Then, we compute the average of $J_{\text{acc}}^{\text{nom}}(\mathbf{G}, \mathbf{Y})$ across the test trajectories, denoted by $\bar{J}_{\text{acc}}^{\text{nom}}$. These results are reported in the second column of Table 4.1. Note that $\bar{J}_{\text{acc}}^{\text{nom}}$ is measured in meters (m) since for its computation only the positions of the car are considered (i.e., the remaining features such as velocity and acceleration are neglected as they can be uniquely computed by the positions). The same applies to all metrics discussed in the rest of this section.

Target Trajectory: Second, we specify the target trajectories $\mathbf{Y}$. The third column in Table 4.1 quantifies how different the target trajectories are from the ground truth one. Formally, we compute the $J_{G-Y}(\mathbf{G}, \mathbf{Y}) = \sum_{m=t+1}^{t+F} \frac{\|\mathbf{G}_m - \mathbf{Y}_m\|_2}{F}$, for each trajectory in the test set, using, again, only the car positions. Then, we compute the average of $J_{G-Y}(\mathbf{G}, \mathbf{Y})$ across the test trajectories, denoted by $\bar{J}_{G-Y}$. These results are reported in the third column of

Table 4.1. The larger the $J_{G-Y}(\mathbf{G}, \mathbf{Y})$ is, the more the desired trajectory deviates from the ground truth.

Evaluation of Attack Success: In the fourth column of Table 4.1, we report the performance of TA4TP as captured by the objective function J in (2.1a). Specifically, we compute (2.1a) for each test trajectory as an input and then we report the average across all trajectories, denoted by $\bar{J}$. The lower the $\bar{J}$ is, the more successful the attack is. In this setup, we terminate the optimization algorithm when either the loss function J is less than 0.02 m or the maximum number of iterations $K_{\max} = 100$ has been reached. Also, we used the Adam optimizer to compute $\mathbf{\Delta}^{k+1}$ (as opposed to gradient descent mentioned in Alg. 1) due to its fast convergence properties. Observe that good prediction accuracy on normal trajectories does not necessarily lead to good adversarial robustness. For instance, Grip++ has a better nominal accuracy in the Apolloscape dataset than Trajectron++ does (see the second column).

However, performance of Grip++ on this dataset seems to be more vulnerable than Trajectron++ to adversarial perturbations (see the fourth column). Also, observe in the fourth column that $\bar{J}$ is quite small implying that the predicted trajectories are sufficiently close to the target ones; see e.g., Figure 4.1.

We note that this may not always be the case depending on the physical constraints and the target trajectory. For example, if the physical constraints are very tight and the target trajectory is rather aggressive (i.e., too far from the nominal prediction), then the optimal perturbed trajectory, as per (2.1), may not achieve a low loss as per (2.1a). This is demonstrated in Fig. 4.2; fooling the DNN model into predicting such target trajectories requires relaxing the physical constraints.

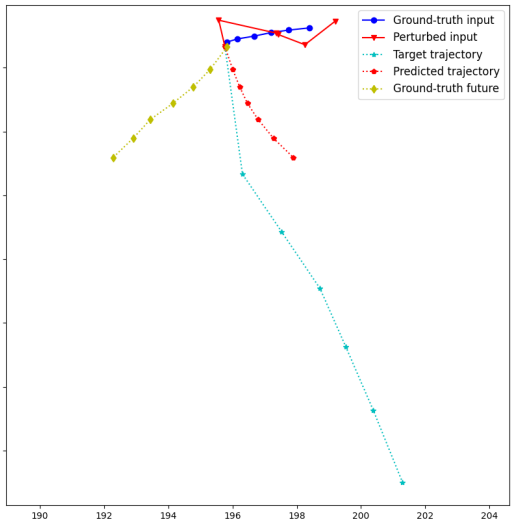

Figure 4.2: Performance of TA4TP given an aggressive target trajectory. The target trajectory requires the car to move along the cyan direction with a velocity that is significantly higher than the one associated with the input path. This difference in the velocity is illustrated by the large distance between the waypoints in the target trajectory.

	$\bar{J}$ (m) [Adam]	T (secs) [Adam]	$\bar{J}$ (m) [grad]	T (secs) [grad]
Grip_apolloscape	0.331	4.699	0.635	4.089
Grip_nuscenes	0.132	3.749	0.196	4.313
Trajectron_apolloscape	0.364	35.790	0.495	35.978
Trajectron_nuscenes	0.077	39.093	0.199	46.777

Table 4.2: Summary of results for TA4TP with $K_{\max} = 10$ and $\tau = 0.02$m.

Attack Design Runtime: The last two columns in Table 4.1 show the average time required to generate an adversarial trajectory using gradient descent (as in (3.1)) and Adam optimizer. As expected, the Adam optimizer is significantly faster than the standard gradient descent method.

Particularly, Adam reduces the computational time by at least 42%. Also, notice that based on the runtimes shown in Table 1, execution of the proposed attack in real time may be prohibitive. To mitigate this, a smaller maximum number of iterations $K_{\max}$ can be selected in Alg. 1 which, however, may compromise the accuracy of the attack (as measured by (2.1a)). For instance, in Table 4.2, we run the same set of experiments as before but with $K_{\max} = 10$. Observe that the runtimes are at least 10 times smaller than the ones reported in Table 1 (where $K_{\max} = 100$). Also, observe in the second column of Table 4.2, that the accuracy of TA4TP has decreased (compared to the one in Table 4.1), but it still achieves a satisfactory performance. In the fourth column of Table 4.2, we also report the corresponding deviation error $\bar{J}$ for the standard gradient descent approach. As expected, the Adam optimizer is faster and achieves a better performance within a fixed number of iterations. We also note that potentially an adversary can design adversarial trajectories offline, store them in a library, and select them online when needed.

DNN Robustness to Random Noise on Clean Inputs: Next, we investigate how/if small random (i.e., non-adversarial) noise affects the performance of the considered DNNs

	$\bar{J}_{\text{acc}}^{\text{nom}}$(m)	$\bar{J}$(m) [Adam]
Grip_apolloscape	0.413	2.260
Grip_nuscenes	1.365	1.219
Trajectron_apolloscape	0.742	2.153
Trajectron_nuscenes	3.710	1.040

Table 4.3: DNN robustness analysis against random noise on clean inputs.

in nominal settings. Particularly, we examine the performance of the DNN models when random noise is embedded in clean (i.e., non-adversarial) inputs. To generate a small amount of random noise, we apply the following process. Given a clean trajectory c, we compute the average distance between consecutive waypoints denoted $\bar{\delta}_c$. Then for each waypoint in c, we sample a new waypoint within a ball centered at the original waypoint with radius $0.02\bar{\delta}_c$. These new waypoints constitute the 'noisy clean' inputs that are close enough to the original ones. Then, given these noisy inputs, we compute the average nominal accuracy $\bar{J}_{\text{acc}}^{\text{nom}}$ and the average deviation $\bar{J}$ from the target paths, as defined before; hereafter, we assume $K_{\max} = 100$ and $\tau = 0.02$ and we consider the same target paths as the ones considered before. The results are reported in Table 4.3 and they should be compared against the ones in the second and fourth column of Table 4.1. Observe that the nominal accuracy $\bar{J}_{\text{acc}}^{\text{nom}}$ of both models has dropped due to the random noise demonstrating their sensitivity; Grip++ seems to be more robust to random noise than Trajectron++ is. Notice also that $\bar{J}$ is significantly high for both models meaning that simply applying random noise cannot fool them into predicting the desired trajectories.

DNN Robustness to Random Noise on Adversarial Inputs: We repeat the same process as above but in adversarial settings. In other words, we examine how the DNN models perform in the vicinity of adversarially crafted trajectories. Specifically, we compute $\bar{J}$ after adding a small amount of noise (exactly as discussed above) into the adversarial inputs generated for Table 4.1. This metric for Grip++ on the Apolloscape and Nuscenes dataset is 0.203 m and 0.253 m, respectively. Similarly, for Trajectron++, we get that the deviation error $\bar{J}$ on the Apolloscape and Nuscenes datasets is 0.507 m and 0.099 m, respectively. This error is close to the corresponding one reported for the 'noiseless' adversarial inputs on the fourth column of Table 4.1. This also implies robustness of TA4TP against random noise that may occur naturally e.g., due to slippery roads or wind gusts. Additionally, by comparing these values with the respective ones for the 'noisy clean' inputs (see the last column in Table 4.3), we see that the DNN models seem to be more robust in the vicinity of adversarial inputs than in the vicinity of clean inputs. We believe that this observation may also be useful to detect adversarial inputs. Similar observations have been used to detect adversarial inputs to image classifiers [22, 25, 30, 34]. Specifically, to detect whether an input image is benign or not, these works investigate how the DNN output changes under transformations (e.g., compression, rotation, or adding noise) applied to the inputs.

Effect of traffic density: Trajectory prediction models model the interaction among objects as a graph structure to enhance prediction performance. To study the factor of traffic density, we perform the following experiment. First we randomly sample 20 test trajectories from the Apolloscape dataset and we compute the average deviation $\bar{J}$, defined earlier, when (i) all agents in the scene are considered versus (ii) all other agents besides the adversary and a randomly selected agent are dropped from the scene. We denote by $\bar{J}_{\text{all}}$ and $\bar{J}_2$ the average deviation $\bar{J}$ in the settings (i) and (ii), respectively. As for Grip++, we get that $\bar{J}_{\text{all}} = 0.215$ m and $\bar{J}_2 = 0.331$m while for Trajectron++, we get that $\bar{J}_{\text{all}} = 0.034$ m and $\bar{J}_2 = 0.102$ m. Observe that the attack remains successful in both settings in the sense that the deviation from the target trajectory is quite low. It is also worth noting that $\bar{J}_{\text{all}} < \bar{J}_2$, i.e., it seems to be 'easier' to fool the DNN models in high traffic density scenarios. Nevertheless, this result may be specific to this experimental setup.

Chapter 5

Some future work attempts

In this section, we decided to try the conformal prediction on detecting the adversarial examples we generated To make the autonomous system safer.

Conformal prediction is a powerful technique for detecting out-of-distribution (OOD) data, which can be leveraged to identify adversarial inputs in machine learning systems. The core concept of conformal prediction is to generate confidence regions for predictions by calibrating them using nonconformity scores, which are computed as a measure of dissimilarity between a new input and a training dataset. In the context of adversarial input detection, the nonconformity scores can be used to identify instances that deviate significantly from the expected data distribution. By setting a predefined significance level, conformal prediction can produce prediction intervals for each input, which allows for the quantification of uncertainty. If the prediction intervals for a particular input are wide or do not contain the predicted class, it is indicative of potential OOD data or an adversarial attack. By carefully selecting the nonconformity measure and significance level, the conformal prediction may have the potential to effectively detect and mitigate the impact of adversarial inputs on the model's performance, thus enhancing the robustness of machine learning systems.

However, due to the length of the sequence of the dataset being too short, the conformal prediction may not be able to make a reliable prediction this time. But we will still try new ways to detect adversarial inputs.

Chapter 6

Conclusion

We proposed TA4TP, the first targeted adversarial attack for DNN models used for trajectory forecasting tasks. We demonstrated experimentally that TA4TP can design input trajectories that look natural and are capable of fooling DNN models into predicting desired outputs. We believe that the proposed method will allow users to evaluate as well as enhance robustness of trajectory prediction DNN models.

References

[1] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *Conference on Robot Learning*, pages 420–429. PMLR, 2020.

[2] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture*, 110, 2020.

[3] 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.

[4] Y. Cao, C. Xiao, A. Anandkumar, D. Xu, and M. Pavone. Advdo: Realistic adversarial attacks for trajectory prediction. In *European Conference on Computer Vision*, pages 36–52. Springer, 2022.

[5] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[6] H. Cheng, M. Liu, L. Chen, H. Broszio, M. Sester, and M. Y. Yang. Gatraj: A graph-and attention-based multi-agent trajectory prediction model. *arXiv preprint arXiv:2209.07857*, 2022.

[7] S.-H. Choi, J.-M. Shin, P. Liu, and Y.-H. Choi. Argan: Adversarially robust generative adversarial networks for deep neural networks against adversarial examples. *IEEE Access*, 10:33602–33615, 2022.

[8] F. Djeumou and U. Topcu. Learning to reach, swim, walk and fly in one trial: Data-driven control with scarce data and side information. In *Learning for Dynamics and Control Conference*, pages 453–466. PMLR, 2022.

[9] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool. Deep interactive motion prediction and planning: Playing games with motion prediction models. In *Learning for Dynamics and Control Conference*, pages 1006–1019. PMLR, 2022.

[10] 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.

[11] J. Fang, F. Wang, P. Shen, Z. Zheng, J. Xue, and T.-s. Chua. Behavioral intention prediction in driving scenes: A survey. *arXiv preprint arXiv:2211.00385*, 2022.

[12] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, S. L. Herbert, S. Wang, A. D. Dragan, and C. J. Tomlin. Confidence-aware motion prediction for real-time collision avoidance1. *The International Journal of Robotics Research*, 39(2-3):250–265, 2020.

[13] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos. Reduced variance deep reinforcement learning with temporal logic specifications. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 237–248, 2019.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[15] L. Hewing, E. Arcari, L. P. Fröhlich, and M. N. Zeilinger. On simulation and trajectory prediction with gaussian process dynamics. In *Learning for Dynamics and Control*, pages 424–434. PMLR, 2020.

[16] M. Hosseinzadeh, B. Sinopoli, and A. F. Bobick. Toward safe and efficient human-robot interaction via behavior-driven danger signaling. *arXiv preprint arXiv:2102.05144*, 2021.

[17] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[18] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 954–960, 2018.

[19] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato. Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *IEEE Transactions on Artificial Intelligence*, 3(2):90–109, 2021.

[20] L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[21] S. Kalluraya, G. J. Pappas, and Y. Kantaros. Multi-robot mission planning in dynamic semantic environments. *arXiv preprint arXiv:2209.06323*, 2022.

[22] Y. Kantaros, T. Carpenter, K. Sridhar, Y. Yang, I. Lee, and J. Weimer. Real-time detectors for digital and physical adversarial inputs to perception systems. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 67–76, 2021.

[23] Y. Kantaros and M. M. Zavlanos. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *The International Journal of Robotics Research*, 39(7):812–836, 2020.

[24] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[25] R. Kaur, S. Jha, A. Roy, S. Park, E. Dobriban, O. Sokolsky, and I. Lee. idecode: In-distribution equivariance for conformal out-of-distribution detection. In *The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, 2022.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] J. Li, F. Schmidt, and Z. Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *International Conference on Machine Learning*, pages 3896–3904. PMLR, 2019.

[28] X. Li, X. Ying, and M. C. Chuah. Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving. *arXiv preprint arXiv:1907.07792*, 2019.

[29] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas. Safe planning in dynamic environments using conformal prediction. *arXiv preprint arXiv:2210.10254*, 2022.

[30] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[31] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[32] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[33] K. Nakamura and S. Bansal. Online update of safety assurances using confidence-based predictions. *arXiv preprint arXiv:2210.01199*, 2022.

[34] F. Nesti, A. Biondi, and G. Buttazzo. Detecting adversarial examples by input transformations, defense perturbations, and voting. *arXiv preprint arXiv:2101.11466*, 2021.

[35] N. Nikhil and B. Tran Morris. Convolutional neural network for trajectory prediction. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[36] M. Omainska, J. Yamauchi, T. Beckers, T. Hatanaka, S. Hirche, and M. Fujita. Gaussian process-based visual pursuit control with unknown target motion learning in three dimensions. *SICE Journal of Control, Measurement, and System Integration*, 14(1):116–127, 2021.

[37] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[38] R. Peddi, C. Di Franco, S. Gao, and N. Bezzo. A data-driven framework for proactive intention-aware motion planning of a robot in a human environment. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5738–5744. IEEE, 2020.

[39] S. Pfrommer, T. Gautam, A. Zhou, and S. Sojoudi. Safe reinforcement learning with chance-constrained model predictive control. In *Learning for Dynamics and Control Conference*, pages 291–303. PMLR, 2022.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[41] 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.

[42] A. Sarkar, J. Feng, Y. Vorobeychik, C. Gill, and N. Zhang. Reward delay attacks on deep reinforcement learning. *arXiv preprint arXiv:2209.03540*, 2022.

[43] J. F. Schumann, J. Kober, and A. Zgonnikov. Benchmark for models predicting human behavior in gap acceptance scenarios. *arXiv preprint arXiv:2211.05455*, 2022.

[44] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15159–15168, 2022.

[45] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora. Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 6(2):2256–2263, 2021.