An Adversarial Attack on DNN-based Adaptive Cruise Control Systems

Abstract

DNN-based Adaptive Cruise Control (ACC) systems are very convenient but also safety critical. Although many prior works have explored physical adversarial attacks on DNN models, those attacks are mostly static and their effects on a real-world ACC system are not clear. In this work, we propose the first end-to-end attack on ACC systems, and we test the safety indication on the state-of-the-art ACC product. The experimental results show that our approach can make the vehicle driving with ACC accelerate unsafely and cause a rear-end collision.

1. Introduction

Adaptive Cruise Control (ACC) is a Level-2 driving automation technology that can automatically adjust the vehicle's velocity based on the velocity and position of the vehicle in front (a.k.a. the lead), to keep pace with the lead, as well as staying at a safe distance from the lead. Due to its high convenience for human drivers, ACC has been available in vehicles from many manufacturers such as Honda, Toyota, Nissan, Chevrolet, Subaru, and Volkswagen [4, 6, 3, 2, 5, 1]. Although convenient, ACC is extremely safety critical since if it makes the wrong driving decision and accelerates/brakes unsafely, it may rear-end the vehicle in front or get rear-ended by the vehicle behind.

The most critical step in ACC is lead detection. To achieve this, most ACC systems use camera as the only or major sensor to perceive the driving environment [4, 6, 3, 2, 5, 1]. Due to its high accuracy, Deep Neural Networks (DNNs) are widely adopted in ACC systems to extract the status of the lead from the image captured by camera. However, many previous works have shown that DNN models are vulnerable to deliberately crafted inputs (a.k.a. adversarial examples). More importantly, recent works have proposed many adversarial examples that can be printed and attached on a real-world object, making some DNN models used in Autonomous Driving (AD) systems malfunction. For example, in [17] and [21], real stop signs with an adversarial poster or sticker on it cannot be detected by object detection models such as YOLO v2 [14]. In [19], after applying a mask on a license plate, this license plate can no longer be recognized by an SSD model. Additionally, in [18], vehicles covered with certain patterns can be hidden from Light-Head RCNN [13]. Although in these works, the proposed adversarial examples were tested under various angles, distances, and illumination conditions. They are all *static* attacks which have only been evaluated with static images, and have not been tested with a product-grade driving system. In a very recent work [16], an end-to-end attack on lane-keeping assistance system is proposed; however, this approach does not apply to ACC systems.

In this work, we design and implement the first approach to attack DNN-based ACC systems. Specifically, 1) to practically and legally introduce perturbations, we add a patch on the back of the lead to pretend to be a mobile advertisement. 2) We develop an effective algorithm to construct a robust physical adversarial patch that can mislead an ACC system for a sufficient number of consecutive camera frames, under various conditions. 3) We test the end-to-end effect of the attack in CARLA [11], a widely-used driving simulator: we import the lead vehicle with a patch on the back into CARLA, and let the vehicle driving with OpenPilot ACC system [7] follow this lead in CARLA. OpenPilot is a state-of-the-art Level-2 AD product. The preliminary results show that our attack can cause a rear-end collision between two vehicles. Anonymized videos of this attack are available at `https://sites.google.com/view/acc-adv/`.

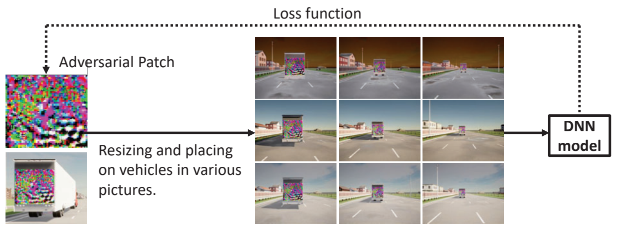


Figure 1. The work flow of adversarial patch generation.

2. Background

ACC is one of the key functions in modern Level-2 AD systems. With ACC, the driver only needs to set the maximum velocity V_{MAX}, (a.k.a. the cruise velocity), and the ACC system automatically adjusts the vehicle velocity within $[0, V_{MAX}]$: if there is a close vehicle (a.k.a. a lead) in front, the system controls the velocity of the ego vehicle based on the position and velocity of the lead. For example, if a slower vehicle moves in front of the ego vehicle, the ACC system will control the ego vehicle to slow down; when the lead accelerates and the velocity is higher than V_{MAX} or when the lead moves out of the way, the ego vehicle will accelerate to achieve V_{MAX}.

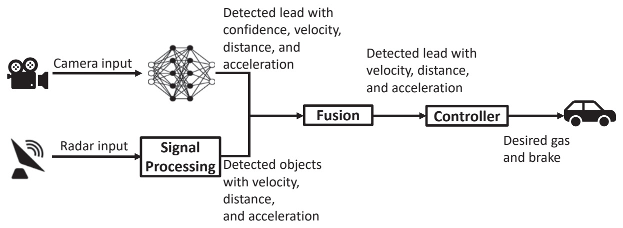

Camera input
Detected lead with confidence, velocity, distance, and acceleration
Detected lead with velocity, distance, and acceleration
Fusion
Controller
Radar input
Signal Processing
Detected objects with velocity, distance, and acceleration
Desired gas and brake

Figure 2. The overview of the ACC system in OpenPilot.

Figure 2 shows an overview of the ACC system design in OpenPilot. This system operates in three steps:

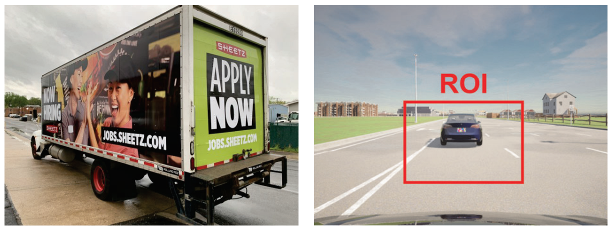
Lead Detection (LD). Lead detection is the most critical step in an ACC system, since the driving decisions are made mainly based on the sensor inputs. Radars are very accurate at detecting and tracking the status of objects (e.g., velocity), but they cannot tell if the detected object is a lead (vehicle). Thus, in OpenPilot and many other recent ACC products [6, 4, 3, 2, 5], both camera and radar sensors are used to perceive the environment. First, as shown in Figure 2, a DNN model is used to extract the lead information from each camera frame[1]. In frame t, the detected lead is associated with the confidence score $s_t \in [0, 1]$, and the status of the lead $y_t^c = (d_t^c, l_t^c, v_t^c, a_t^c)$, where d_t^c and l_t^c are the distances between the lead and ego vehicle along the driving direction and lateral direction; v_t^c and a_t^c are the velocity and acceleration of the lead (relative to the ego vehicle) along the driving direction; we use superscript c, r for camera and radar (mentioned later). Second, traditional signal processing is used to extract the information of the objects detected by radar. The i-th object in frame t can be represented as $y_t^{r,i} = (d_t^{r,i}, l_t^{r,i}, v_t^{r,i}, a_t^{r,i})$.

Fusion. The detection results from camera and radar are fused to get the final result. The detailed fusion process can be found in [7]. In short, 1) if $s_t < 0.5$, this detected lead is treated invalid; 2) if $s_t > 0.5$, y_t^c is compared with each $y_t^{r,i}$ in frame t to find the best match.

Longitudinal control and vehicle actuation. In each frame t, the fused status of a valid lead ($y_t^* = (d_t^*, l_t^*, v_t^*, a_t^*)$) is sent to a Model Predictive Control (MPC) [15] module to calculate the desired velocity and acceleration for the next step. Then, the desired velocity and acceleration are used by a Proportional-Integral (PI) control loop [12] to control the gas and brake of the vehicle. With a detected lead, the control objective is to smoothly achieve the same velocity with the lead as well as maintaining the safe distance with the lead. The safe distance is defined as:

$$\begin{aligned} d_t^{safe} &= v_t^{ego} * TR - (v_t^{lead} - v_t^{ego}) * TR \\ &+ v_t^{ego} * v_t^{ego}/(2G) - v_t^{lead} * v_t^{lead}/(2G) + 4. \end{aligned} \tag{1}$$

The safe distance in frame t, d_t^{safe}, is related to the absolute velocity of the lead and ego vehicle (v_t^{lead} and v_t^{ego}). It is the distance the ego vehicle needs if the lead starts braking at G $\mathrm{m/s^2}$ and TR seconds later the ego vehicle starts braking too at G $\mathrm{m/s^2}$ and is able to eventually stop at 4 m away from the lead. In OpenPilot v0.8.3, $TR = 1.8$ and $G = 9.81$.



(a) (b)

Figure 3. (a) A truck with advertisement on the side and back, and (b) an example of ROI area.

3. Adversarial Attack on ACC

3.1. Threat Model and Problem Formulation

We assume an attacker who aims to attack the ACC system on the victim vehicle to cause a rear-end collision between the victim vehicle and the lead in front of it. The attacker can achieve this by manipulating objects in the physical world to "fool" the DNN model in the ACC system. We assume that the attacker can possess the same ACC system as the one on the victim vehicle and has full knowledge of the ACC algorithm by either reverse-engineering the algorithm or accessing the code repository if it is open-source.

Physical approach. We focus on the scenario in which the attacker tries to launch the attack by placing an adversarial patch on the back of a vehicle (i.e. an adversarial vehicle), and cause the vehicle behind which is driving with ACC to rear-end the adversarial vehicle. We choose this attack scenario for two reasons:

R1: **Feasibility.** In many ACC systems, a Region of Interest (ROI) filtering is performed to the raw camera frame before the DNN model is applied (Figure 3(b)). This is to extract the most import area to improve the accuracy of lead detection [20]. Thus, modifying objects out of the ROI area such as the surrounding buildings will not affect the lead detection.

R2: **Legality.** Within the ROI area, some previous works proposed to manipulate the license plate of the

[1]Though rare, OpenPilot can detect multiple leads when there are multiple vehicles in front in the same lane with the ego vehicle. Here we only consider the case in which one lead is detected since other leads are usually obstructed by the closest lead.

lead [19] or the patches on the road [16] to achieve the attack. However, such approaches can be illegal in many cities/countries. In contrast, modifying the appearance of a vehicle is more likely legitimate: 1) if the attacker owns the adversarial vehicle, then he is allowed to change the vehicle appearance (without covering the light, license plate, etc.) according to the law in many places; 2) the attacker can also use a vehicle that has advertisement on the back as the adversarial vehicle. As shown in Figure 3(a), many truck companies offer the back of the truck as a place for advertisement. Thus, the attacker can control the back appearance of a truck legally by advertising.

Attack incentives. We consider two possible incentives: *business competition* and *automotive insurance fraud*. First, one ACC system provider may build an attack to cause an accident on a vehicle driving with a rival company's ACC system, and thus damage its reputation as well as gaining unfair competitive advantages. Second, automotive insurance frauds happen very frequently in many countries [9]. Adversarial attacks can be a new way to perform the fraud: the attacker can get compensation from the insurance company if the vehicle driving with ACC behind him is affected by the patch and rear-ends the attacker's vehicle.

3.2. Attack Algorithm

Our goal is to make a vehicle driving with an ACC system (ACC vehicle for short) accelerate unsafely and eventually cause a rear-end collision. As shown in Section 2, when driving at a velocity less than V_{MAX} in frame t, an ACC vehicle will accelerate when one of the two following cases occurs: 1) $s_t < 0.5$; 2) $s_t > 0.5$, $d_t^c > d_t^{safe} + \lambda$. λ is caused by the fusion process and is about $0.25 * d_t^c$. Thus, different than previous attacks on object detection models that only focus on reducing s_t to cause a disappearance attack, we include both s_t and d_t^c in our objective to learn adversarial examples. And later in Section 4 we will show that this change is very important to the attack.

Sequential attack. To have end-to-end impact on an ACC system, the attack needs to affect a sufficient number of consecutive camera frames. In fact, the attacks on later frames depend on the ones on earlier frames. For example, if the adversarial patch successfully causes the lead to be not detected or to be considered far away in frame t, then the ACC system will control the vehicle to accelerate, and the patch (and also the lead) will become larger in the image captured in frame t+1. Thus, to cause a collision on an ACC vehicle which is following a lead with a following distance D_0 (i.e. the distance between the lead and the ACC vehicle), we need an adversarial patch that is robust when the following distance lies in $[0, D_0]$. To achieve this, we use Expectation over Transform (EOT) [10] in our attack.

Specifically, given the digital template of the patch on the lead x (e.g., the advertisement patch), we add perturbation δ on it to generate the adversarial patch $x^{'} = x + \delta$. Then, to apply EOT, we simulate the images taken by the camera under different conditions: we collect real input images from a driving simulator which are taken with the following distance lying in $[0, D_0]$, under different weathers and backgrounds, and rescale the adversarial patch to place it on the vehicles in the sampled images, as shown in Figure 1. The collected images can be denoted as

$$\begin{aligned} x' &= x + \delta \\ z_i &= T(x', f_i) \end{aligned} \tag{2}$$

where T denotes the overall process of rescaling and placing the adversarial patch, f_i is the i-th sampled image.

3.3. Objective

The overall loss function is a combination of the adversarial loss and the Total Variation (TV) loss:

$$\mathcal{L} = \underbrace{\mathcal{L}_{conf} + \lambda_1 \mathcal{L}_{dis}}_{\mathcal{L}_{adv}} + \lambda_2 \mathcal{L}_{TV}. \tag{3}$$

We consider the following mathematical formulation for the adversarial loss:

$$\begin{aligned} \mathcal{L}_{adv} &= \mathcal{L}_{conf} + \lambda_1 \mathcal{L}_{dis} \\ &= \mathbb{E}_{z_i \in Z} - \log(1 - s(z_i)) - \lambda_1 |d(z_i)/d^f(f_i)| \end{aligned} \tag{4}$$

where $s(z_i)$ and $d(z_i)$ denote the confidence score of the detected lead and the detected distance between the lead and the ego vehicle, with the input image z_i; $d^f(f_i)$ denotes the following distance when the image f_i was taken. Z is the set of simulated images. We use binary cross entropy to minimize the confidence score of the detected lead, and we add the detected distance in the loss to maximize it.

In addition to the adversarial loss, we use TV loss as done in [17], to smooth the perturbation pattern and make it continuous, in case that the printing machine cannot accurately distinguish pixelated patterns. The TV loss of the adversarial patch can be formulated as:

$$\mathcal{L}_{TV} = \sum_{i,j} |\delta_{i+1,j} - \delta_{i,j}| + |\delta_{i,j+1} - \delta_{i,j}| \tag{5}$$

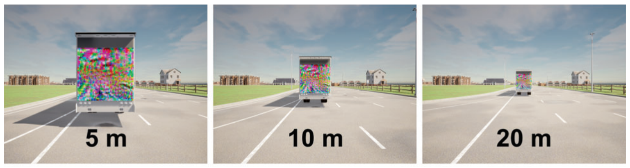


Figure 4. The adversarial patch used in the experiments.

4. Evaluation

We evaluate our method on the state-of-the-art open-source Level-2 AD system, OpenPilot [7], since it has been shown to have better performance than products from many other manufacturers [8]. The ACC system of OpenPilot is introduced in Section 2. To test the end-to-end security/safety effect of our attack, we perform the evaluation in a driving simulator named CARLA [11]. Specifically, we first "drive" a vehicle (controlled by OpenPilot) in CARLA following a lead (a box truck) to collect some input images (Step 1). Then, based on these images, we learn the adversarial patch following the methodology in Figure 1 (Step 2). Next, we modify the vehicle appearance in CARLA to place the adversarial patch on the back of the lead (Step 3). Finally, we "drive" the OpenPilot vehicle again to follow the lead with the adversarial patch on its back to test the attack (Step 4).

In Step 4 above, we focus on a scenario where the lead is slowing down. There are *three phases* in this scenario:

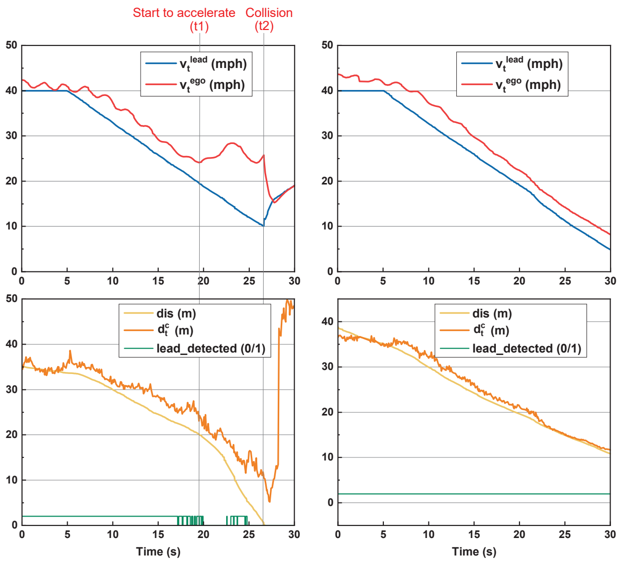
P1: The ACC vehicle is following the lead as normal at a constant velocity which is 30-80 mph, and the following distance is thus 28-70 m.

P2: The lead starts to slow down due to some special situation in front of it. Later, the ACC vehicle starts to slow down accordingly, and it will get closer to the lead (because d_t^{safe} is smaller with lower velocity).

P3: When the distance between the two vehicles is smaller than a threshold (D^*), the adversarial patch starts to take effect and the ACC vehicle starts to accelerate until hitting the lead.

To achieve the above attack, we train the adversarial patch to be robust when the following distance lies in $[0, D^*]$. D^* is 20 m in our experiments. We choose this attack scenario due to its good stealthiness. When the lead is at a high velocity or it just starts to slow down, the driver of the ACC vehicle may pay full attention to ensure that ACC is working correctly. However, after the driver notices that his vehicle is taking the right action (i.e. slowing down accordingly), he will very likely be relaxed/distracted and fully trusting the ACC system. Then, when the vehicle suddenly stops slowing down and starts to accelerate, it may take the driver longer than normal to react and take control.

Early Results. For the loss function, we set $\lambda_1 = 0.01, \lambda_2 = 0.001$. The adversarial patch is shown in Figure 4. Figure 5 shows the details of the ACC vehicle's behavior when the lead is slowing down from 40 mph, with/without the adversarial patch. Here we assume that the speed limit of the road is 45 mph, i.e. the driver sets V_{MAX} to be 45 mph. At the beginning when $t < 5$ s, the ACC vehicle is following the lead at ~40 mph as normal. When $t = 5$ s, the lead starts slowing down smoothly to eventually stop, and in the benign case, the ACC vehicle will also slow down and eventually stop at ~4 m away from the lead. In contrast, in the attack case, although the ACC vehicle slows down at the beginning, when $t = 17$ s and the following distance is 22.5 m, the patch takes effect and the ACC vehicle starts to accelerate at $t1 = 19.5$ s and crashes into the lead at $t2 = 26.6$ s. The velocity of the lead increases after $t2$ because it is pushed by the ACC vehicle.



(a) Attack (b) Benign

Figure 5. The detailed process of the attack and benign case, where dis denotes the following distance, dis_detected is a binary which represents if $s_t > 0.5$ is true.

As shown in Figure 5(a), from 22.5 s to 24.5 s the attack on s_t is not very robust due to the variations on illumination, background, and other conditions, and the ACC vehicle thus starts to slow down. However, since we also attack d_t^c and make it much larger than the real following distance, the ACC vehicle does not slow down dramatically although there is a detected lead. d_t^c is not as sensitive to the mentioned variations as s_t because the DNN in OpenPilot is very robust on s_t. We do not only attack d_t^c because the planner limits the acceleration when a lead is detected.

5. Concluding Remarks and Future Plans

In this work, we design and implement the first end-to-end attack on real-world ACC systems. We evaluate our approach on a state-of-the-art ACC system and the results have shown that our attack can successfully cause a rear-end collision on a vehicle driving with ACC. In the future, we plan to perform more comprehensive evaluations in variant driving scenarios and on more ACC systems. In addition, we plan to propose a defense for this attack. Potential defense mechanisms include 1) a better fusion algorithm, 2) more sensor inputs (e.g., lidar), and 3) adversarial training.

References

[1] Adaptive cruise control (acc). https://www.volkswagen-newsroom.com/en/adaptive-cruise-control-acc-3664.

[2] Helping drivers automatically keep their distance from the vehicle ahead. https://my.chevrolet.com/how-to-support/safety/adaptive-cruise-control.

[3] Honda adaptive cruise control information overview. https://www.germainhonda-annarbor.com/adaptive-cruise-control-information/.

[4] Intelligent cruise control. nissan-global.com/EN/TECHNOLOGY/OVERVIEW/icc.html.

[5] Subaru eyesight. https://www.subaru.com/engineering/eyesight.html.

[6] Toyota safety sense (TSS). https://www.buyatoyota.com/home/tools/toyota-safety-sense/.

[7] OpenPilot: Open source driving agent. https://github.com/commaai/openpilot, 2018.

[8] We hit the road with comma.ai's assisted-driving tech at ces 2020. https://www.cnet.com/roadshow/news/comma-ai-assisted-driving-george-hotz-ces-2020/, 2018.

[9] 6 types of car insurance fraud. https://www.bankrate.com/insurance/car/fraud, 2021.

[10] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[12] Tore Hägglund. An industrial dead-time compensating pi controller. *Control Engineering Practice*, 4(6):749–756, 1996.

[13] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017.

[14] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[15] Jacques Richalet, André Rault, JL Testud, and J Papon. Model predictive heuristic control. *Automatica (journal of IFAC)*, 14(5):413–428, 1978.

[16] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jack Jia, Xue Lin, and Qi Alfred Chen. Hold tight and never let go: Security of deep learning based automated lane centering under physical-world attack. *arXiv preprint arXiv:2009.06701*, 2020.

[17] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[18] Tong Wu, Xuefei Ning, Wenshuo Li, Ranran Huang, Huazhong Yang, and Yu Wang. Physical adversarial attack on vehicle detector in the carla simulator. *arXiv preprint arXiv:2007.16118*, 2020.

[19] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Tsung-Yi Ho, and Yier Jin. Beyond digital domain: Fooling deep learning based recognition system in physical world. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1088–1095, 2020.

[20] Sibel Yenikaya, Gökhan Yenikaya, and Ekrem Düven. Keeping the vehicle on the road: A survey on on-road lane detection systems. *ACM Comput. Surv.*, 46(1), July 2013.

[21] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1989–2004, 2019.