2021 IEEE Intelligent Vehicles Symposium (IV)
July 11-17, 2021. Nagoya, Japan

End-to-end Uncertainty-based Mitigation of Adversarial Attacks to Automated Lane Centering

Ruochen Jiao[1,2], Hengyi Liang[1,2], Takami Sato[3], Junjie Shen[3], Qi Alfred Chen[3], Qi Zhu[2]

Abstract— **In the development of advanced driver-assistance systems (ADAS) and autonomous vehicles, machine learning techniques that are based on deep neural networks (DNNs) have been widely used for vehicle perception. These techniques offer significant improvement on average perception accuracy over traditional methods, however have been shown to be susceptible to adversarial attacks, where small perturbations in the input may cause significant errors in the perception results and lead to system failure. Most prior works addressing such adversarial attacks focus only on the sensing and perception modules. In this work, we propose an end-to-end approach that addresses the impact of adversarial attacks throughout perception, planning, and control modules. In particular, we choose a target ADAS application, the automated lane centering system in OpenPilot, quantify the perception uncertainty under adversarial attacks, and design a robust planning and control module accordingly based on the uncertainty analysis. We evaluate our proposed approach using both public dataset and production-grade autonomous driving simulator. The experiment results demonstrate that our approach can effectively mitigate the impact of adversarial attack and can achieve $55\% \sim 90\%$ improvement over the original OpenPilot.**

I. Introduction

Machine learning techniques have been widely adopted in the development of autonomous vehicles and advanced driver-assistance systems (ADAS). Most autonomous driving and ADAS software stacks, such as Baidu Apollo [1] and OpenPilot [2], are generally composed of four-layered modules: sensing, perception, planning, and control [3]. The sensing and perception modules collect data from the surrounding environment via a variety of sensors such as cameras, LiDAR, radar, GPS and IMU, and use learning-based perception algorithms to process the collected data and understand the environment. The planning and control modules leverage the perception results to propose a feasible trajectory and generate detailed commands for the vehicle to track the trajectory. In those systems, deep neural networks (DNNs) are widely used for sensing and perception in transportation scenarios [4]–[6] , such as semantic segmentation, object detection and tracking, as they often provide significantly better average perception accuracy over traditional feature-based methods. For planning and control, there are also increasing interests in applying neural networks with techniques such as reinforcement learning and imitation learning [7], due to their capabilities of automatically learning a strategy within a complex environment.

[1] These two authors contribute equally to the work.
[2] Ruochen Jiao, Hengyi Liang and Qi Zhu are with the Department of Electrical and Computer Engineering, Northwestern University, IL, USA.
[3] Takami Sato, Junjie Shen and Qi Alfred Chen are with the Department of Computer Science, University of California, Irvine, CA, USA.

However, the adoption of DNN-based techniques in ADAS and autonomous driving also brings significant challenges to vehicle safety and security, given the ubiquitous uncertainties of the dynamic environment, the disturbances from environment interference, transient faults, and malicious attacks, and the lack of methodologies for predicting DNN behavior [8]. In particular, extensive studies have shown that DNN-based perception tasks, such as image classification and object detection, may be susceptible to adversarial attacks [9], [10], where small perturbations to sensing input could result in drastically different perception results. There are also recent works on attacking DNN-based perception in ADAS and autonomous vehicles by adding adversarial perturbations to the physical environment in a stealthy way [9], [11]–[13]. For instance, [12] generates a dirty road patch with carefully-designed adversarial patterns, which can appear as normal dirty patterns for human drivers while leading to significant perception errors and causing vehicles to deviate from their lanes within as short as 1 second.

The prior works addressing adversarial attacks mostly focus on detecting anomaly in the input data [14], [15] or making the perception neural networks themselves more robust against input perturbations [16], [17]. In ADAS and autonomous driving, however, the impact of adversarial attacks on system safety and performance is eventually reflected through vehicle movement, taking into account of planning and control decisions. Thus, we believe that for those systems, it is important to take a holistic and end-to-end approach that addresses adversarial attacks throughout the sensing, perception, planning and control pipeline.

In our preliminary work recently published in a work-in-progress paper [18], we studied the automated lane centring (ALC) system in OpenPilot [2], a popular open-source ADAS implementation, and investigated how the dirty road patch attack from [12] could affect perception, planning and control modules. We discovered that a *confidence score* generated by the perception module could serve as a sensitive signal for detecting such attack, however it does not quantitatively measure the extent of the attack and cannot be effectively used for mitigation. In this paper, motivated by the findings from [18], we propose a novel end-to-end approach for *detecting and mitigating* the adversarial attacks on the ALC system. Our approach quantitatively estimates the *uncertainty* of the perception results, and develops an adaptive planning and control method based on the uncertainty analysis to improve system safety and robustness.

In the literature, methods have been proposed to address the uncertainties of various modules in the ADAS

2021 IEEE Intelligent Vehicles Symposium (IV) | 978-1-7281-5394-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/IV48863.2021.9575549

and autonomous driving pipeline. For instance, the method proposed in [19] utilizes estimated uncertainty as a threshold to decide which sensor is reliable. In the OpenPilot implementation, Multiple Hypothesis Prediction [20] is utilized to estimate the prediction confidence. Some works propose methods to detect out-of-distribution inputs [21] and design probabilistic deep learning based perception models [22] and planning models [23]. Different from these prior methods, our approach takes a system-level view and addresses the uncertainty from adversarial attacks throughout sensing, perception, planning and control. While this work focuses on the dirty road patch attack, we believe that our methodology can be applied to other adversarial attacks that cause perception uncertainties, and may be extended to address more general uncertainties (e.g., those caused by environment interference or transient faults). Specifically, our work makes the following contributions:

- We analyzed the impact of dirty road patch attack across the ADAS pipeline, and developed a method to quantitatively measure the perception uncertainty under attack, based on the analysis of both model and data uncertainties in the perception neural network.
- We developed an uncertainty-aware adaptive planning and control method to improve system safety and robustness under adversarial attacks.
- We conducted experiments on both public dataset and LGSVL [24], a production-grade autonomous driving simulator. The results demonstrate that our approach can significantly improve the system robustness over the original OpenPilot implementation when under adversarial attacks, reducing the deviation of lateral deviation by $55\% \sim 90\%$.

The rest of the paper is organized as follows. Section II introduces the ALC system in OpenPilot and the adversarial attack model to this system. Section III presents our uncertainty-based mitigation approach to address such adversarial attacks. Section IV shows the experimental results.

II. ALC System and Adversarial Attacks

The Automated Lane Centering (ALC) system, one of the Level 2 autonomous driving systems, are widely deployed in modern commercial vehicles. In the ALC system, the perception module collects vision and distance input from cameras and radars, and outputs the perception of the environment to the planning and control module, which generates a desired trajectory and controls vehicle steering and acceleration. In the following, we will take the open-source software Openpilot (Fig. 1) as an example to illustrate ALC's architecture.

A. DNN-based Perception Module

Recently, DNN-based models achieve the state-of-the-art performance in lane detection tasks [25] and are widely adopted in production-level ALC systems today such as Tesla AutoPilot [26] and OpenPilot. The DNN-based perception module can detect lane lines and objects and provide necessary information for the planning and control module.

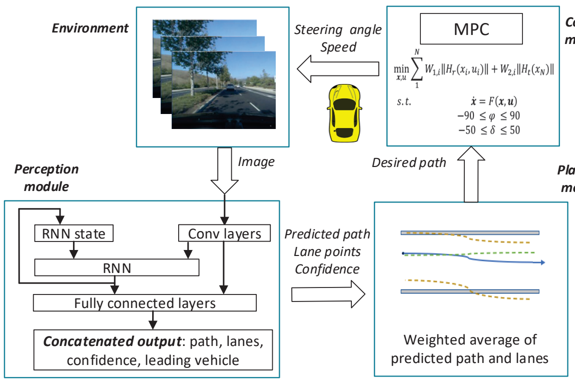


Fig. 1: OpenPilot pipeline: the ALC system consists of a DNN-based perception module, a trajectory planning module, and an MPC controller.

In ALC tasks, since the lane lines are continuous across consecutive frames, recurrent neural network (RNN) is typically applied to utilize the time-series information to make the prediction more stable [27]. A convolutional neural network (CNN) processes the current frame and feeds the combination of the CNN's and RNN's outputs into a fully-connect network. The concatenated output includes a predicted path (an estimation of the path to follow based on perception history), the detected right and left lane lines, and corresponding probabilities to indicate the detection confidence of each lane line. Note that the lane detection model first predicts lane points and then fits them into polynomial curves in post-processing for denoising and data compression.

B. Planning and Control Modules

The planning module generates a trajectory for the vehicle to follow. Sampling based methods, model based methods and deep learning models are often applied to the trajectory planning [28]. For ALC, the desired path generated by the planner should be located in the middle between the left lane line and the right lane line. The planner module in OpenPilot generates an estimation of this desired path by calculating a weighted average of the detected left lane line, right lane line and predicted path, with the weights being the confidence scores outputted by the perception module. Intuitively, if the perception module is less confident on the predicted lanes, the generated desired path relies more on the predicted path; otherwise, it will be closer to the weighted average of the predicted left lane line and right lane line [2], [18].

Given the desired path to follow, the lower-level controller calculates the vehicle maneuver and generates commands to control throttle, brake, and steering angle. In OpenPilot ALC, model predictive control (MPC) [29] is used to calculate the steering angle based on simplified system dynamics, vehicle heading constraints, and maximum steering angle constraints. MPC-based approach permits high-precision planning and a certain degree of robustness. For longitudinal control, the acceleration is generated by a PID controller by setting an appropriate reference speed.

C. Attack Model

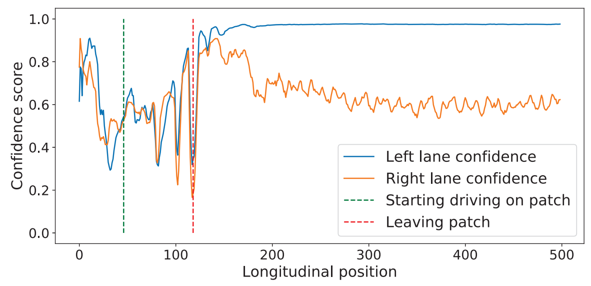
In this paper, we assume a similar attack model as prior work [12]. We focus on attacks achieved through external physical world. In particular, we assume that the attacker cannot hack through software interfaces nor modify the victim vehicle. However, the attacker can deliberately change the physical environment that is perceived by the on-board sensors of the victim vehicle (e.g., cameras). Furthermore, we assume that the attacker is able to get knowledge of the victim's ADAS/autonomous driving system and can drive the same model vehicle to collect necessary data. This can be achieved by obtaining a victim vehicle model and conducting reverse engineering, as demonstrated in [30], [31]. The attacker's goal is to design the appearance of certain object (in our case, a dirty patch) on the road such that 1) the adversarial object appears as normal/seemingly-benign for human drivers, and 2) it can cause the ADAS/autonomous driving system to deviate from the driver's intended trajectory. Some example attack scenarios are discussed in [11], [12]. The work in [11] generates an adversarial billboard to cause steering angle error. [12] generates a gray scale dirty patch on the road such that vehicles passing through the patch will deviate from its original lane, which is the first attack systematically designed for the ALC system and reaches state-of-art attack effect on production-level ADAS. Through our experiments, we will use the dirty road patch attack [12] as a case study, but we believe that our approach can be extended to other similar physical environment attacks. These physical attacks typically will render abnormal behavior in the perception output and then propagate through the entire pipeline.

III. Our End-to-end Uncertainty-based Mitigation Approach for Adversarial Attacks

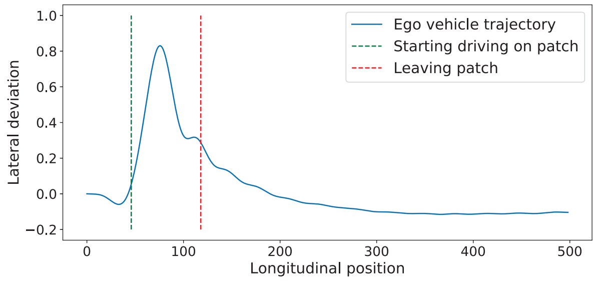
As shown later in Fig. 3, the perception module in our approach involves two neural network models. The original OpenPilot perception model is used in the normal operation mode (i.e., when the overall prediction confidence is high), where it outputs predicted path, lanes and confidence scores. When anomaly is detected, a new perception neural network is used to estimate model uncertainty while generating perception output. The trajectory planner will take the uncertainties into consideration and generate a desired path that is less affected by the adversarial attack. Correspondingly, in the lower-level control, more conservative and uncertainty-aware constraints are used in the MPC and a speed adaptation method is applied to ensure safety. The details of our proposed approach are introduced below.

A. Perception Confidence as Signal of Attack

Measuring the confidence of the DNN's prediction is a significant challenge. In different perception tasks, various methods are applied to estimate the perception confidence. For instance, in YOLO [32], intersection over union (IOU) measures the confidence of regression. In OpenPilot's neural network, a multiple hypotheses prediction (MHP) [20] classifier is trained with cross entropy loss and its output



(a)



(b)

Fig. 2: The patch starts around 40 meters and is 96 meters long. (a) shows that the confidence of the prediction drops, (b) plots the lateral deviation of the ego vehicle.

can represent how confident the lane line is predicted correctly [33].

To investigate how the perception confidence affects the system safety, we conduct a series of experiments with different settings of the attacks As indicated in Fig. 2a, we observe a general phenomenon that the confidence score of the perception module drops significantly when the vehicle is approaching the dirty patch, while the confidence score keeps in a relatively stable level in benign cases or under ineffective noises. Fig. 2 also shows the consistency between the drop in confidence score and the vehicle's lateral deviation under attack. As discussed in Section II, the desired path generated by OpenPilot's path planner can be considered as a weighted average of the predicted left lane, predicted right lane, and the predicted path, with the confidence scores as weights. Under the impact of the dirty patch, confidence of the predicted two lanes drops, resulting in more weights on predicted path. However, the predicted path deviates from the middle of the line eventually, and the desired path leans towards wrong directions.

Based on such observations, we think that the perception confidence is an interface between the perception and the following planning and control modules: it can both indicate whether perception module is under adversarial attack and influence the planned trajectory. Our mitigation strategy leverages this interface to switch between different perception modules and applies adaptive planning and control accordingly.

B. Uncertainty Estimation and Safety Bound

While the confidence scores generated by the OpenPilot perception module *qualitatively* show the existence of the

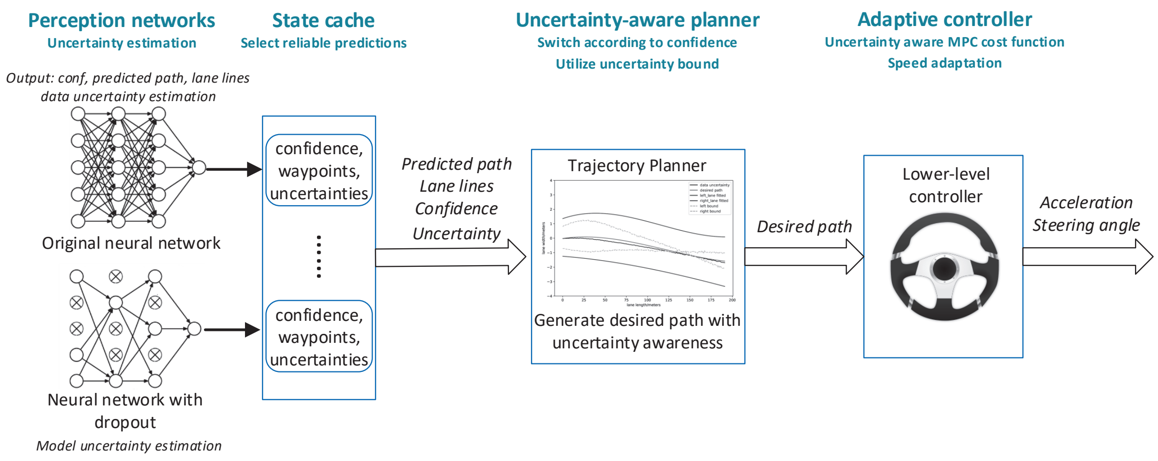

Perception networks
Uncertainty estimation
State cache
Select reliable predictions
Uncertainty-aware planner
Switch according to confidence
Utilize uncertainty bound
Adaptive controller
Uncertainty aware MPC cost function
Speed adaptation
Output: conf, predicted path, lane lines
data uncertainty estimation
Original neural network
Neural network with dropout
Model uncertainty estimation
confidence, waypoints, uncertainties
confidence, waypoints, uncertainties
Predicted path
Lane lines
Confidence
Uncertainty
Trajectory Planner
Generate desired path with uncertainty awareness
Desired path
Lower-level controller
Acceleration
Steering angle

Fig. 3: Overview of our proposed ALC pipeline with uncertainty-based mitigation for adversarial attacks.

attack, they do not *quantitatively* measure the extent of the attack and cannot be effectively used for mitigation. Thus, we developed a new approach for quantifying the perception *uncertainty* that considers both data uncertainty and model uncertainty. Data uncertainty is caused by the input noise/disturbance and intrinsic randomness of the real data while model uncertainty results from a lack of training data in certain areas of the input domain and the test example being out of distribution. By assuming that the data uncertainty follows a normal distribution, we can estimate it by using maximum likelihood approach. In neural networks, the model can take mean negative log-likelihood function as a loss function in the training to capture the aleatory uncertainty [34]. For the perception module, x is the image input and y is the ground truth of left and right lane lines. In Equation (1) below, $\hat{\mu}$ is the predicted mean value and the standard deviation $\hat{\sigma}$ is the estimated data uncertainty:

$$\mathcal{L}(x,y)=\frac{\log\hat{\sigma}^2_{data}(x)}{2}+\frac{(y-\hat{\mu}(x))^2}{2\hat{\sigma}^2_{data}(x)} \quad (1)$$

To estimate model uncertainty, similarly to the Bayesian approaches, we consider that there are distributions over the weights of neural networks and the parameters of the distribution are trained on the training dataset. The weight distribution after training can then be written as $p(\omega \mid \mathrm{X}, \mathrm{Y})$ where X, Y represent input and corresponding target. We apply the Monte Carlo methods to estimate the distribution by adding dropout layers to sample weights with dropout rate Φ [35]:

$$p(\omega|\mathbf{X},\mathbf{Y})\approx \mathrm{Bern}(\omega;\Phi) \quad (2)$$

Therefore, the model uncertainty can be estimated as [36]:

$$\sigma^2_{model}=\frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{\mu}_t-\overline{\boldsymbol{\mu}})^2 \quad (3)$$

The total variance can be derived by adding the model variance σ^2_{model} and the data variance σ^2_{data}, as shown in Equation (4). Since the data uncertainty is always assumed to follow the normal distribution, we can adjust the data uncertainty interval based on different scenarios. To the best of our knowledge, current commercial driving assistance systems do not take the perception uncertainty into consideration.

$$\sigma^2_{total}=\sigma^2_{model}+\sigma^2_{data} \quad (4)$$

When the overall prediction confidence drops below a threshold, we deem that the ego vehicle is experiencing significant environment noise or under adversarial attack. Instead of using the original prediction result, we leverage the obtained prediction uncertainties to more conservatively measure the bounds of left lane and right lane. Let μ_i be the predicted mean of the i-th point of the lane and $\sigma_{total,i}$ be the corresponding error, then $[\mu_i-\sigma_{total,i},\mu_i+\sigma_{total,i}]$ can be considered as the range where the "true" lane point may reside. Specifically, we use $\mu_i-\sigma_{total,i}$ as the lower bound of left lane if μ_i is a point on the left lane while $\mu_i+\sigma_{total,i}$ as the upper bound if it is a point on the right lane. We use a polynomial function $p(i)$ to approximate the points along the lane. The problem can be cast as a weighted least square fitting. For left lane, we fit the curve by minimizing $\sum_i^N w_i|p(i)-(\mu_i-\sigma_{total,i})|$; for right lane, we minimize $\sum_i^N w_i|p(i)-(\mu_i+\sigma_{total,i})|$. Here, w_i is the weight. Since points further away tend to exhibit larger uncertainties, we assign closer points with larger weights by setting $w_i=\frac{1}{\sigma_{data}}$. Fig. 6 shows the bounded predicted lanes using our approach. As we can see, the bounded prediction is more conservative compared to the original approach. In most cases, the bounded left lane and right lane tend to intersect at some point.

C. Uncertainty-aware Trajectory Planner

In the original design of OpenPilot, the desired path can be considered as a weighted sum of the left lane, right lane and predicted path (generated by the perception module). Lanes/path with higher confidence score will be assigned with larger weights and have more impact on the desired path. However, we find out that this approach cannot handle adversarial scenarios. Consider that the attacker can manipulate sensor data such that the confidence scores of the predicted left lane and right lane are much lower while the predicted path is bent towards left. In this case, the desired path will also lean towards left even though the vehicle is driving on a straight road. We argue that, when the confidence score drops below certain threshold, we should explicitly consider the uncertainties instead of just relying on the predicted road curves. That is, we use the uncertainty-bound area derived above to constrain vehicle's speed and steering angle.

The pseudo-code to calculate the desire path in our uncertainty-aware approach is shown in Algorithm 1. The overall confidence lr_{conf} is calculated by considering the confidence of each lane, i.e., $lr_{conf} = l_{conf} + r_{conf} - l_{conf} * r_{conf}$. Lines 5-6 calculate the accumulated uncertainties along the left lane and right lane. ω_{left} and ω_{right} are the corresponding weights assigned to the bounded left lane and bounded right lane. $p_{weighted}$ is the weighted path and the final desired path is the weighted average of $p_{weighted}$ and $p_{openpilot}$ (the latter is the desired path obtained by running the original OpenPilot). The intuition is that if the current prediction has low overall confidence, we will rely more on the uncertainty-aware bounded prediction to mitigate the adversarial attack.

Algorithm 1 DesiredPath: Uncertainty-aware Desired Path Calculation

```
Require: Polynomial for lane lines: $p_l, p_r$;
    the overall confidence of the prediction: $lr_{conf}$
 1: if total confidence $lr_{conf}$ is less than $Conf\_Threshold$ then
 2:     use the original Openpilot to generate the desired path $p_{openpilot}$
 3:     return $p_{openpilot}$
 4: else
 5:     $left_{sum} = \sum_{i=1}^{n} \sigma_{left,i}$
 6:     $right_{sum} = \sum_{i=1}^{n} \sigma_{right,i}$
 7:     $total_{sum} = left_{sum} + right_{sum}$
 8:     $\omega_{left} = \frac{right_{sum}}{total_{sum}}$
 9:     $\omega_{right} = \frac{left_{sum}}{total_{sum}}$
10:     $p_{weighted} = \omega_{left} * p_l + \omega_{right} * p_r$
11:     $desired\ path = (1 - lr_{conf}) * p_{weighted} + lr_{conf} * p_{openpilot}$
12: end if
13: return $desired\ path$
```

Besides utilizing the estimated uncertainty to bound the safe trajectory area and produce the desired path, we also make use of the temporal locality of the path prediction. We notice that the perception and planning modules can produce a safe desired path for about 100 meters with frequency of 20 Hz while the vehicle will only move forward up to several meters in the period. Therefore, the information of consecutive frames has considerable locality and relevance. We maintain a state cache to store the perception output of most recent k consecutive frames. In our experiment, we pick $k = 7$ to store the information of the past 0.35 seconds. In case of adversarial scenarios, the system will select the perception output with highest confidence score from the state cache as the planner input. Taking advantage of the locality, the system robustness to short-term inference will be improved.

D. Adaptive Controller

1) Uncertainty-aware cost function: An MPC controller is used to generate an appropriate steering command based on the ego vehicle status and the desired path. We modify the MPC controller by explicitly considering the more conservative uncertainty-aware bounded lanes. The optimization objective can be written in the form of the summation of a running cost and a terminal cost:

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{i=1}^{N} \mathbf{W}_{1,i} \|\mathbf{H}_r(x_i, u_i)\|^2 + \mathbf{W}_{2,i} \|\mathbf{H}_t(x_N)\|^2 \qquad (5)$$

where $\mathbf{W}_{1,i}$ and $\mathbf{W}_{2,i}$ are weight matrices; $\mathbf{H}_r$ is the reference function to capture the difference between current ego vehicle states and the desired path. $\mathbf{H}_t$ is a measurement function regarding the ego vehicle states at the end of prediction horizon. Intuitively, given the desired path, we want the vehicle to drive along the reference path, but we also want the vehicle to driven on the center of traffic lanes. This is achieved by considering the the distance errors with respect to the desired lane and traffic lanes. However, we argue that the distance error regarding the left lane and right lane should adopt the bounded prediction instead of the original OpenPilot prediction. Specifically, the reference function is written as:

$$\mathbf{H}_r(x_i, u_i) =$$
$$\begin{bmatrix} p_d(x_i) - y_i & e^{-(p_l(x_i) - y_i)} & e^{p_r(x_i) - y_i} & \epsilon_h & \epsilon_a & u^2 \end{bmatrix}^T$$

where p_d is the polynomial representing the desired path, p_l and p_r are fitted polynomial representing bounded left and bounded right lane respectively. ϵ_h and ϵ_a are the error regarding the ego vehicle heading and angular rate respectively. u^2 is a penalty term to avoid aggressive steering. The measurement function $\mathbf{H}_t$ is similar as $\mathbf{H}_r$, except that it does not consider angular rate and penalty. The constraints include 1) system dynamics 2) vehicle heading is limited to $[-90°, 90°]$ and 3) the maximum steering angle is 50°.

2) Speed Adaptation: Generally, the uncertainty for further-way points tends to be considerably large (e.g., Fig. 5) even for benign driving scenario. This means that, for safety consideration, the ego vehicle is not sure about the road structures that are far away. To prevent the vehicle from driving too fast under uncertain scenarios, we apply an emergency brake to the vehicle if its current speed is too fast. Specifically, when the overall prediction confidence drops below the threshold $Conf\_Threshold$, a deceleration α_{max} is applied. The entire end-to-end pipeline of our approach is shown in Algorithm 2. Lines 1-3 calculate prediction uncertainties. Then the result is pushed into state cache. If the confidence of the prediction is too low, we pick the perception result with the highest confidence score from the state cache (line 5-7). Then the bounded predicted lane curves are used to calculate the desired path as in Algorithm 1. Moreover, if the confidence score drops below the threshold, we apply an emergency brake to slow down the vehicle. Here, v_{min} is the minimum speed required. Finally, the MPC controller calculates the steering angle for the next period (line 12).

Algorithm 2 Our Uncertainty-aware pipeline for ALC

```
Require: Current speed $v_{current}$, reference speed $v_{ref}$ and input image $Input$
1: $lr_{conf}, pts_{lane}, \sigma^2_{data} = \boldsymbol{NN}(Input)$
2: $\sigma^2_{model} = \boldsymbol{Monte\_Carlo\_Dropout}(Input)$
3: $\sigma^2_{total} = \sigma^2_{model} + \sigma^2_{data}$
4: push $lr_{conf}, pts_{lane}$ into $state\_cache$
5: if $lr_{conf} < Conf\_Threshold$ then
6:     $pts_{lane}, lr_{conf} = \arg\max_{lr_{conf}}(state\_queue)$
7: end if
8: $desired\_path = \boldsymbol{DesirePath}(pts_{lane}, lr_{conf}, \sigma^2_{total})$
9: if $lr_{conf} < Conf\_Threshold$ then
10:     $v_{ref} = max(v_{current} - \alpha_{max} * \Delta t, v_{min})$
11: end if
12: $steering\_angle, acc = \boldsymbol{MPC}(desired\_path, v_{ref})$
13: return $steering\_angle, acc$
```

IV. Experimental Results

We implement our design within the open-source driving assistance system OpenPilot. We test our proposed approach using both open dataset (comma2k19 dataset [37]) and synthetic scenarios with an autonomous driving simulator.

For the open dataset , we adopt the kinematic bicycle model [38] as the motion model for the ego vehicle. The kinematic bicycle model is commonly used to simulate how vehicles move according to speed and the front steering angle. The kinematic model can produce the vehicle trajectory and obtain the lateral deviation to measure the effectiveness of our proposed mitigation strategy. Moreover, as the trajectory calculated by the motion model may deviate from the original trace, we adopt the motion model based input generation from [12] that combines motion model with perspective transformation to dynamically calculate camera frame updates due to attack-influenced vehicle control.

For synthetic scenarios, we combine OpenPilot with LGSVL [24] to set up a closed-loop simulation environment. LGSVL is a Unity-based multi-robot simulator developed by LG Electronics America R&D Center. We reuse the LGSVL-OpenPilot bridge [12] to transfer sensor data and driving command between the LGSVL simulator and OpenPilot. For both simulation methods, we test the original OpenPilot and ofur proposed design under different attacks as well as in benign situation.

A. Adversarial Attacks

In our work, as mentioned in previous sections, we applied the optimization-based physical attack in [12]. The attack works by placing an optimized patch on the road and it can lead the vehicle out of the lane within 1 second (which means the lateral deviation is larger than 0.735m in highway). The attack against OpenPilot shows high success rate and significant attack effect. To analyze the safety of the system and evaluate our proposed design, we conduct experiments under different settings (perturbation area, stealthiness levels, etc). The benign and attacked inputs are shown in Fig. 4.

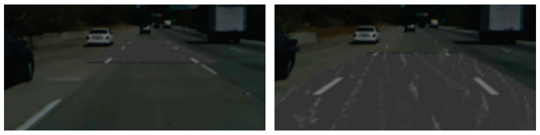


Fig. 4: Input images in benign scenario and under attack.

B. Results for Uncertainty Estimation

For each input frame, our system will calculate the data uncertainty by distributional parameter estimation and model uncertainty by Monte Carlo Dropout methods. The distributional parameter estimation is embedded in the original OpenPilot perception neural network. For the Monte Carlo Dropout, we only activate dropout in inference. There is always a trade-off between inference speed and estimation accuracy: larger number of samples results in higher accuracy and longer time. According to the analysis in [36] and our experiments, we set the dropout rate as 0.2 and the number of samples as 20. In this setting, the system can obtain precise variance estimation and reach a processing frequency of about 20Hz. The system will estimate uncertainties for 192 points of the predicted lane line in every frame. In experiments, we find that in most cases, data uncertainty and model uncertainty are at roughly the same order of magnitude (left subgraph in Fig. 5). The uncertainty is relatively large when the vehicle is approaching the adversarial patch, which is consistent with the observations in confidence estimation. An example for our uncertainty estimation is shown in Fig. 5. In a single frame, the uncertainties increase with the distance from the current position. This observation facilitates the planner to estimate a safe bound. As shown in Fig. 6, the uncertainty bound will form a triangle-like safe area and we will get a desired path from our planner, which is less affected by the attacks.

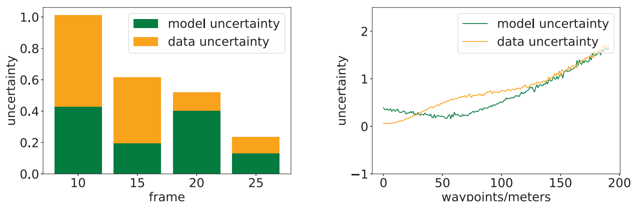


Fig. 5: Data uncertainty and model uncertainty from selected and representative frames. The left figure shows the overall uncertainties in different frames. The right figure shows the uncertainties increase with the distance in one frame.

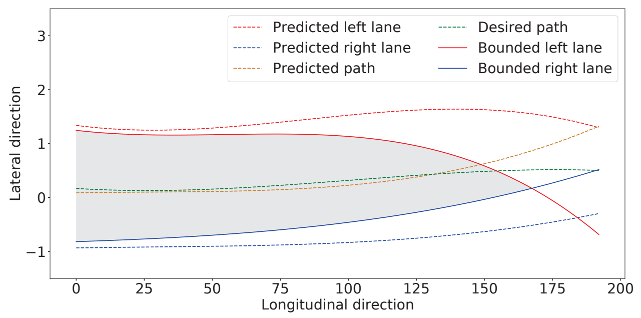


Fig. 6: The shaded gray area indicates the safe area bounded by our estimation of uncertainties. The bounded area and the desired path don't deviate to the left under attacks.

C. Mitigation Results for Open Dataset

We first test our proposed pipeline with highway image dataset and kinematic model. The input is consecutive image frames captured in a straight four-lane highway and the vehicle's speed is 20 m/s. In the benign case, our proposed systems and the original OpenPilot can both drive in the center of the lane. Fig. 7 shows that the adversarial attacks can lead the vehicle with the original OpenPilot out of road and the deviation is more than 1.5 meters. In contrast, the system with our proposed uncertainty estimation and adaptive uncertainty-aware planner and controller can significantly mitigate the attack's effect, reducing the maximum lateral deviation by about 66.8%. Besides, the experiments specifically demonstrate that the adversarial attack's effect can be further mitigated by utilizing the temporal locality with state cache. However, in this simulation setting, it is difficult for us to change the vehicle's speed and perception sampling frequency since they were determined by the driving scenario and sensor configurations in the dataset, which motivates us to conduct the following closed-loop simulation in LGSVL.

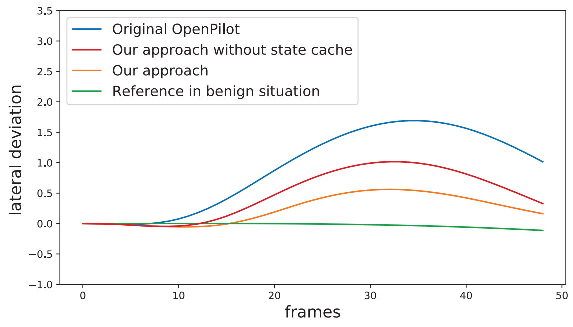


Fig. 7: Comparison of our proposed method (with or without state cache) with the original OpenPilot and reference benign situation in terms of lateral deviation. The y-axis positive direction means left.

D. Mitigation Results in LGSVL-OpenPilot Environment

In the closed-loop LGSVL simulation, we set the vehicle to follow a reference speed (20m/s) and control the throttle using a PID controller. Fig. 8 compares our proposed approach with the original OpenPilot under adversarial scenario. The baseline is the solid blue curve which is the trajectory of the original OpenPilot driving under benign scenario. The solid green curve is the trajectory of the original OpenPilot driving under adversarial scenario where the patch is placed at 40 meters. As we can see, the original OpenPilot does not take prediction uncertainties under consideration and can deviate significantly under adversarial attack. Our proposed is also affected by the patch on the road but can significantly alleviate the adversarial impact.

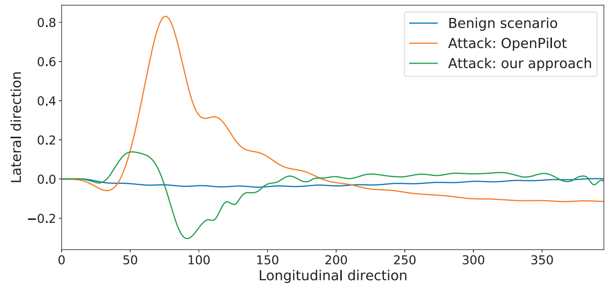


Fig. 8: Ego vehicle trajectory under different scenarios.

To further evaluate our proposed approach, we conduct the experiments with different settings of the patch. We place the patch on different positions along the road (40m, 80m and 120m along the longitudinal direction) and adapt the perturbation area ratio (PAR) of the patch. The improvement of our proposed approach over the original OpenPilot ALC is shown in table I. The perturbation area ratio denotes the percentage of the pixels on the patch that are allowed to be perturbed. For our experiment, we generate dirty road patch with PAR ranging from 25% to 100%. The maximum lateral deviation varies depending on different experiment settings. Throughout our experiments, the lateral deviation of the original OpenPilot ALC ranges from 0.8 to 1.2 meters, which is large enough to drive the victim vehicle out of the lane boundary. In contrast, the lateral deviation of our proposed approach ranges from 0.1 to 0.4 meter (still within the lane boundary). Overall, our approach can reduce the lateral deviation by $55.34\% \sim 90.76\%$, under various patch settings.

TABLE I: Improvement of our proposed approach over the original OpenPilot ALC, tested in LGSVL-OpenPilot environment under different experiment settings: the perturbation area ratio (PAR) and the patch's position along the longitudinal direction.

PAR (%) \ Position (m)	40	80	120
100	59.33%	66.96%	76.15%
75	66.44%	90.76%	82.24%
50	55.34%	76.72%	71.93%
25	61.07%	66.07%	65.74%

V. Conclusions

In this work, we proposed a novel end-to-end uncertainty-based mitigation approach for adversarial attacks to the automated lane centering system. Our approach includes an uncertainty estimation method considering both data and model uncertainties, an uncertainty-aware trajectory planner, and an uncertainty-aware adaptive controller. Experiments on public datasets and a production-grade simulator demonstrate the effectiveness of our approach in mitigating the attack effect. We believe that our methodology can be applied to other ADAS and autonomous driving functions, and will explore them in future work.

Acknowledgment

We gratefully acknowledge the support from NSF grants CNS-1839511, CNS-1834701, IIS-1724341, CNS-1850533, CNS-1929771, CNS-1932464, USDOT grant 69A3552047138 for CARMEN UTC (University Transportation Center), and ONR grant N00014-19-1-2496.

References

[1] "Baidu apollo." [Online]. Available: https://apollo.auto/index.html

[2] comma ai openpilot: an open source driver assistance system. [Online]. Available: https://github.com/commaai/openpilot/

[3] E. Yurtsever, J. Lambert, A. Carballo, *et al.*, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[4] H. Gao, B. Cheng, J. Wang, *et al.*, "Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4224–4231, 2018.

[5] K. Xu, X. Xiao, J. Miao, *et al.*, "Data driven prediction architecture for autonomous driving and its application on apollo platform," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 175–181.

[6] M. Siam, S. Elkerdawy, M. Jagersand, *et al.*, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 1–8.

[7] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," *arXiv preprint arXiv:1810.06544*, 2018.

[8] Q. Zhu, W. Li, H. Kim, Y. Xiang, *et al.*, "Know the unknowns: Addressing disturbances and uncertainties in autonomous systems : Invited paper," in *ICCAD*, 2020, pp. 1–9.

[9] K. Eykholt, I. Evtimov, E. Fernandes, *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[10] S.-T. Chen, C. Cornelius, J. Martin, *et al.*, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 52–68.

[11] H. Zhou, W. Li, Z. Kong, *et al.*, "Deepbillboard: Systematic physical-world testing of autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 347–358. [Online]. Available: https://doi.org/10.1145/3377811.3380422

[12] T. Sato, J. Shen, N. Wang, Y. Jia, X. Lin, and Q. A. Chen, "Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security '21)*, 2021.

[13] C. Sitawarin, A. N. Bhagoji, A. Mosenia, *et al.*, "Darts: Deceiving autonomous cars with toxic signs," 2018.

[14] K. Lee, K. Lee, H. Lee, *et al.*, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 7167–7177.

[15] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[17] A. Madry, A. Makelov, L. Schmidt, *et al.*, "Towards deep learning models resistant to adversarial attacks," 2019.

[18] H. Liang, R. Jiao, T. Sata, *et al.*, "Wip: End-to-end analysis of adversarial attacks to automated lane centering systems," *to appear in the Third International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2021.

[19] R. Nakashima and A. Seki, "Uncertainty-based adaptive sensor fusion for visual-inertial odometry under various motion characteristics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3119–3125.

[20] C. Rupprecht, I. Laina, R. DiPietro, *et al.*, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3591–3600.

[21] F. Cai and X. Koutsoukos, "Real-time out-of-distribution detection in learning-enabled cyber-physical systems," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020, pp. 174–183.

[22] L. Sun, W. Zhan, and M. Tomizuka, "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2111–2117.

[23] C. M. Hruschka, M. Schmidt, D. Töpfer, *et al.*, "Uncertainty-adaptive, risk based motion planning in automated driving," in *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE, 2019, pp. 1–7.

[24] G. Rong, B. H. Shin, H. Tabatabaee, *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," 2020.

[25] Z. Wang, W. Ren, and Q. Qiu, "Lanenet: Real-time lane detection networks for autonomous driving," *arXiv preprint arXiv:1807.01726*, 2018.

[26] S. Ingle and M. Phute, "Tesla autopilot: semi autonomous driving, an uptick for future autonomy," *International Research Journal of Engineering and Technology*, vol. 3, no. 9, pp. 369–372, 2016.

[27] Q. Zou, H. Jiang, Q. Dai, *et al.*, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE transactions on vehicular technology*, vol. 69, no. 1, pp. 41–54, 2019.

[28] X. Qian, I. Navarro, A. de La Fortelle, *et al.*, "Motion planning for urban autonomous driving using bézier curves and mpc," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Ieee, 2016, pp. 826–833.

[29] M. Bujarbaruah, X. Zhang, H. E. Tseng, *et al.*, "Adaptive mpc for autonomous lane keeping," *arXiv preprint arXiv:1806.04335*, 2018.

[30] "Experimental security research of tesla autopilot," 2019. [Online]. Available: https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf

[31] S. Checkoway, D. McCoy, B. Kantor, *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Security Symposium*, 2011.

[32] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[33] D. Ramos, J. Franco-Pedroso, A. Lozano-Diez, *et al.*, "Deconstructing cross-entropy for probabilistic binary classifiers," *Entropy*, vol. 20, no. 3, p. 208, 2018.

[34] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *arXiv preprint arXiv:1703.04977*, 2017.

[35] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[36] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153–3160, 2020.

[37] H. Schafer, E. Santana, A. Haden, *et al.*, "A commute in data: The comma2k19 dataset," 2018.

[38] J. Kong, M. Pfeiffer, G. Schildbach, *et al.*, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1094–1099.