machine learning & knowledge extraction

MDPI

Article

Reconstruction-Based Adversarial Attack Detection in Vision-Based Autonomous Driving Systems

Manzoor Hussain and Jang-Eui Hong *

Software Intelligence Engineering Lab, Department of Computer Science, Chungbuk National University, Cheongju 28644, Republic of Korea; manzoorhussain@chungbuk.ac.kr
* Correspondence: jehong@chungbuk.ac.kr

Abstract: The perception system is a safety-critical component that directly impacts the overall safety of autonomous driving systems (ADSs). It is imperative to ensure the robustness of the deep-learning model used in the perception system. However, studies have shown that these models are highly vulnerable to the adversarial perturbation of input data. The existing works mainly focused on studying the impact of these adversarial attacks on classification rather than regression models. Therefore, this paper first introduces two generalized methods for perturbation-based attacks: (1) We used naturally occurring noises to create perturbations in the input data. (2) We introduce a modified square, HopSkipJump, and decision-based/boundary attack to attack the regression models used in ADSs. Then, we propose a deep-autoencoder-based adversarial attack detector. In addition to offline evaluation metrics (e.g., F1 score and precision, etc.), we introduce an online evaluation framework to evaluate the robustness of the model under attack. The framework considers the reconstruction loss of the deep autoencoder that validates the robustness of the models under attack in an end-to-end fashion at runtime. Our experimental results showed that the proposed adversarial attack detector could detect square, HopSkipJump, and decision-based/boundary attacks with a true positive rate (TPR) of 93%.

Keywords: deep learning; adversarial attacks; robustness; safety; autonomous vehicles; autoencoders

Citation: Hussain, M.; Hong, J.-E. Reconstruction-Based Adversarial Attack Detection in Vision-Based Autonomous Driving Systems. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1589–1611. https://doi.org/10.3390/make5040080

Academic Editor: Vasile Palade

Received: 17 September 2023
Revised: 27 October 2023
Accepted: 2 November 2023
Published: 7 November 2023

1. Introduction

The advancement in deep neural network (DNN) models used in autonomous driving systems (ADSs) is rapidly transforming transportation systems. Generally, ADSs are complex modular systems that divide the driving task into submodules such as perception, localization, planning, and control. However, in recent studies, many researchers have explored the potential of end-to-end deep-learning models for ADSs. These models directly map the raw pixels into steering commands, such as NVIDIA's ADS model [1]. Recent studies have shown that end-to-end models are remarkably successful, especially for imitation learning that learns to imitate human behaviors. The perception system in modular and monolithic models heavily depends on advanced deep-learning models to understand the surrounding environments.

Recently, the robustness of DNN models at training and testing time has received attention from academia and industries. The pioneer researchers in the field of adversarial attacks mainly studied the impact of adversarial attacks on classification models [2–4], while the impact of these attacks on regression models remained unexplored. Like any other DNN model, the models used in the perception system of ADSs are also vulnerable to adversarial attacks. These adversarial attacks are specially crafted by perturbing the original input data (i.e., input image). Such a minor perturbation causes incorrect scene understanding, which may cause a fatal crash in ADSs. Safety is the ultimate goal of ADSs, and it heavily depends upon the overall robustness of the DNN model used in the perception module. Thus, correctly understanding the surrounding environment leads to

correct trajectory planning [5]. This means that the model must be resilient to any minor perturbation in the input data that could otherwise lead to unsafe trajectory planning. Consequently, these adversarial attacks raise challenges and concerns regarding the safety and robustness of DNN models used in safety-critical applications like ADSs.

The primary objective of the adversarial attacks is to take into account the model's vulnerabilities and craft adversarial input to fool the DNN model into producing incorrect results [5–8]. Various adversarial attack strategies have recently been proposed to fool the DNN model into producing incorrect results in different application domains [9–14]. Among these attack strategies, DeepSearch [15], DeepFool [16], projected gradient descent (PGD) [17], and the fast gradient sign method (FGSM) [2] are commonly adopted methods for adversarial attacks. Evaluating the impact of adversarial attacks on classification models is straightforward. An adversarial attack is considered successful if the model under attack misclassifies (i.e., it generates an incorrect label) the adversarial input with high confidence [18]. However, evaluating the impact of adversarial attacks on the regression model is challenging as we need to quantify the resulting deviation due to the attack in the final output. For example, a minor deviation in the steering angle caused by a perturbation attack may be considered to have resulted in an unsuccessful attack because the minor deviation does not affect the safety of ADSs. Therefore, to consider an attack successful, the adversarial attack must lead to a higher deviation in the final output.

Commonly used methods to attack vision-based ADSs are perturbation and physical attacks (i.e., patch attack) [19–22]. In perturbation-based attacks, a minor change in the pixel values is carried out in the input image, while in the patch attacks, the pixel values are affected at large. Many researchers have proposed different methods to fool the DNN models used in ADSs; for example, [23] proposed a method that places physically realizable adversarial objects on the top of vehicles to make them invisible to object detectors. Placing the adversarial objects on top of the target vehicle makes it invisible to the LiDAR-based object detector. Another researcher [24] proposed perturbation- and patch-attack-based adversarial attacks on the object detection systems of ADSs and successfully fooled the RCNN and DSGN models. PhysGAN [25] proposed a unique method to generate physical-world-resilient adversaries to mislead the perception systems of ADS.

On the other hand, various studies have proposed methods to detect and defend against adversarial attacks. These methods can be broadly categorized as gradient-based methods [26], in which the gradient information of the DNN model is used to detect adversarial attacks. Denoising methods [27,28] and reconstruction-based methods [29] use different reconstruction methods, such as autoencoders. On the other hand, statistical methods [30] use statistical distances such as the energy distance (ED) and maximum mean discrepancy (MMD) to detect adversaries, and, lastly, the most popular approach to detect and defend against adversarial attacks is ensemble-based methods [31], in which multiple models are trained with various adversarial training techniques to detect and defend against adversarial examples.

Although adversarial attack techniques and their impact on DNN models have been widely studied recently, current studies mainly focus on adversarial attacks and their impact on classification tasks. The impact of these attacks on regression models, particularly in end-to-end models used on ADSs, remains unexplored. Additionally, existing approaches are domain-specific, and they lack generalizability. For example, in the case of physical attacks (i.e., patch attacks), it is difficult to cast the patch on the target object with the correct location and color. Patch attacks must be crafted in the correct color and location on the target object; thus, in many cases, these attacks are mostly not successful. In most cases, physical attacks are more attention-grabbing and noticeable, which is against the basic principle of adversarial attacks. Other types of attacks, such as white box, gray box, and model extraction, require extensive domain knowledge and full access to the DNN model; therefore, these methods are almost impossible to apply to the ADS domain.

In addition to these demerits, existing studies measure the effectiveness of the attack strategies based on the decline in accuracy (i.e., misclassification in the case of classification

tasks) or the rate of increased loss of the model under attack. However, a decline in model performance does not necessarily cause driving hazards in the case of ADSs. Moreover, the existing approaches for detecting and defending against the adversarial attack only evaluate the performance of these attacks in an offline fashion using different performance metrics such as the accuracy, precision, true positive rate (TPR), false positive rate (FPR), and F1 score. However, achieving a higher precision, TPR, FPR, or F1 at the offline evaluation stage does not guarantee that the adversarial attack detection system will perform well at runtime. Therefore, a set of metrics would need to evaluate the impact of these attacks and the performance of the attack detection system in an offline, as well as online, fashion. Typically, any adversarial attack aims to maximize the loss of the model under attack, which, ultimately, decreases the accuracy or performance. It is easy to evaluate the impact of the attack on the classification model by quantifying (i.e., maximizing the loss implies a lower model performance) the misclassification rate. However, in the case of the regression model used in end-to-end ADSs, it is hard to evaluate the impact of these attacks.

In order to fill the research gap between the performance of the model under attack and the robustness of the DNN model in ADSs, this article first proposed perturbation-based attack strategies on vision-based ADSs. Then, it introduces a novel reconstruction-based adversarial attack detection method using a deep autoencoder, as shown in Figure 1. Finally, in addition to offline evaluation metrics, we propose a set of online evaluation metrics to evaluate the impact of adversarial attacks and the performance of the adversarial attack detector. Our online evaluation metrics take into account the reconstruction loss of the deep autoencoder to evaluate the impact of these attacks and the performance of the attack detector rather than the decline in accuracy or increased loss of the model due to these attacks. Our study considers end-to-end vision-based ADSs that rely on camera-based perception for environmental sensing. The end-to-end based ADS we consider in this paper is the NVIDIA-DAVE-2 model, which directly maps the input image from the camera sensor into the driving parameters. The adversarial attack detector compares the reconstruction loss of the deep autoencoder against a threshold to flag whether the input image is perturbed due to adversarial attacks.

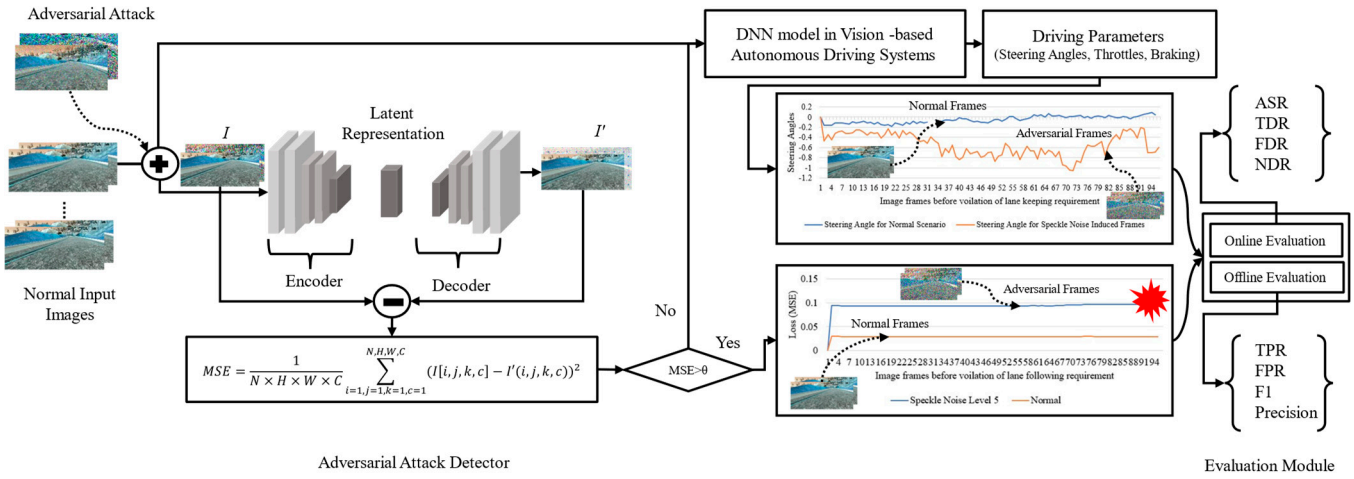

Figure 1. A framework for detecting adversarial attacks on DNN model in vision-based ADSs.

To acquire comprehensive results from our experimental setup, we adopted two types of methods to create perturbations. In the first method, we apply four different types of naturally occurring noises—*shots, Gaussian, impulse, and speckle noise*—with different attack intensities that can occur either naturally due to sensor degradation or by perturbing the input image through adversarial attacks, while, in the second method, we adopted modified square, HopSkipJump, and decision-based adversarial attacks to create perturbations. In addition to offline evaluation metrics, we also measure the rate of deviation in the vehicle trajectory, attack success rate based on reconstruction loss, successful attack detection rate,

and the normal driving rate at runtime to verify the robustness of the DNN model under attack. We summarize the contributions of this article as follows:

(1) We propose an autoencoder-based adversarial attack detection method that considers the reconstruction loss of the autoencoder to flag whether the input image is adversarial or normal. Our attack detector flags an image as adversarial if its reconstruction loss exceeds the set threshold. A lower reconstruction loss indicates normal images. In comparison, a higher reconstruction loss indicates adversarial images.
(2) To evaluate the robustness of the DNN model used in ADSs under adversarial attacks, an end-to-end evaluation framework is proposed, which takes the reconstruction loss of the deep autoencoder and produces the score for evaluation metrics such as the attack success rate (ASR), true detection rate (TDR), false detection rate (FDR), and normal driving rate (NDR). These additional metrics will allow us to evaluate the robustness of the model under attack at runtime.
(3) Through extensive experimental analyses, we observed the increasing and decreasing trends in the reconstruction loss of deep autoencoders for normal and adversarial images and its effect on newly proposed metrics for evaluating the robustness of the model under attack. Our extensive experiment confirmed that even state-of-the-art models like NVIDIA-DAVE-2 also lack robustness against perturbation attacks.
(4) Our experimental results, in terms of quantitative and qualitative analysis, also confirm that the deep-autoencoder-based adversarial attack detection system effectively detects adversaries with high accuracy.

The remainder of the paper is organized as follows: We introduce the literature review on adversarial attacks in Section 2, and Section 3 discusses our attack strategies and proposed methods for detecting these adversarial attacks, followed by a detailed discussion of experimental results in Section 4, and we conclude our paper in Section 5.

2. Related Work

In this section, we describe the existing literature from the perspectives of adversarial attacks on vision-based ADSs, end-to-end ADSs, and methods to detect and defend against adversarial attacks on ADSs.

2.1. Adversarial Attacks on Autonomous Driving Systems

Among the adversarial attacks, perturbation attacks are the most commonly used attacks to fool the DNN models in the domain of ADSs [32]. According to the general survey on adversarial vulnerabilities of ADSs, perturbation and physical attacks are the most dangerous threats to vision-based ADSs as they directly manipulate the input images [33]. An adversarial attack aims to make a small manipulation in the input images to increase the errors in the final output of the target models. Commonly used methods are gradient- and optimization-based approaches to create perturbations in the input images. Based on these methods, researchers have developed different adversarial attacking techniques. The authors of [34] used the fast gradient signed method (FGSM) technique to generate incorrect traffic signs and mislead the DNN-based traffic sign detection system. Li et al. [22] presented a square attack technique to generate a perturbation in the input image to fool the DNN model and misclassify the input images. A novel technique to generate the 3D adversarial patterns in LiDAR point cloud perturbation was proposed by the authors of [20]. The adversarial examples generated by their method successfully deceive the 3D object detector. The study of Li et al. [19] presented a method that perturbed the trajectories of the target vehicle to generate incorrect trajectories that ultimately cause driving hazards. In this study, the authors perturbed the trajectory of ADSs instead of attacking the raw point cloud of the LiDAR sensors. Similarly, He et al. [35] also presented a method to perturb the LiDAR's point cloud for adversarial attacks on the LiDAR sensing module. Cao et al. [36] presented a technique that generates the adversarial point based on optimization-based adversarial attacking methods. It generates a fabricated point cloud to deceive the 3D object detection model used in ADSs. Nassi et al. [37] used a projector to

project misleading traffic signs, which caused the ADSs to recognize the deceptive signs as real. Similarly, many other researchers focused on different modules to attack, such as road sign recognition [38,39], image recognition [40,41], object detection [42,43], and traffic sign recognition [44–46]. Other than the adversarial attacks mentioned above, false data injection [47] and denial of service attacks [48] are well-known attacks on autonomous driving systems. Zhao et al. [49] proposed a method to detect false data injection attacks in connected and automated vehicles through cloud-based sandboxing, while, on the other hand, Hosseinzadeh et al. [50] worked on detecting active attacks in constrained cyber–physical systems. This technique is based on analyzing the input and output data. Their approach focused on preventing actuation attacks in which the attacker controls the data exchange between the controllers and actuators.

Various researchers have also considered adversarial attacks on end-to-end models and studied the impact of these attacks through extensive experiments. The authors of [51] presented a method to attack end-to-end models by replacing roadside billboards with adversarial fabricated billboards. Their experimental results showed that the adversarially fabricated billboards could significantly deviate the steering angles from the original steering angle. The study of the authors in [52] presented a simple, physically realizable attack by painting a black line on the road. Their study also considered end-to-end models and extensively explored the vulnerabilities of such models using their proposed attacking technique. The systematic investigation from their study showed their adversarial attacking technique successfully deviates the vehicle trajectory toward adversarially created lines. A physical attack method called PhysGAN was proposed by the authors of [25], which generates a realistic billboard similar to the original one to fool the model used in ADSs. The adversarially generated billboard could successfully deviate the vehicle trajectory from the original route. They targeted the end-to-end ADS model, revealing that the adversarial billboard causes a significant deviation in steering angle. However, none of the above-mentioned methods directly studied the impact on the robustness of the target end-to-end ADS model and the behavior of ADSs under adversarial attacks. Only a few studies, such as [24], studied the impact of these attacks on the safety of ADSs. However, their experimental setup focuses more on object detection rather than the direct impact on vehicle behavior. For example, their studies focused more on the impact of patch and perturbation attacks on two state-of-the-art object detection models such as RCNN and DSGN. Instead of studying the impact on other modules, such as traffic sign recognition, object detection, or roadside sign recognition, we are more focused on studying the impact of these attacks on the robustness of the target DNN model in an end-to-end setting.

2.2. End-to-End Autonomous Driving Systems

The end-to-end ADS model learns to map the raw input sensor data into the vehicle control parameters. In contrast to the modular approach, where driving tasks are subdivided into smaller tasks such as perception, scene understanding, planning, and control, the end-to-end models work in a monolithic approach. End-to-end models are trained through either imitation learning [53,54] or reinforcement learning [55,56]. The pioneering work in the field of imitation learning called Autonomous Land Vehicle in a Neural Network (ALVINN) was proposed by Pomerleau [57]. In this work, the authors used fully-connected shallow neural networks to perform end-to-end lane following where there was no obstacle. Later, researchers like Eraqi et al. proposed a vision-based end-to-end model for controlling the steering angle, considering temporal dependencies using long short-term memory recurrent neural networks (LSTM) [58]. Recently, more efficient end-to-end models were developed by engineers from NVIDIA based on convolutional neural networks that directly map the raw pixels from input images into steering angles [1]. Other researchers [59,60] also presented a similar idea for controlling the ADS using end-to-end models.

2.3. Methods for Detecting and Defending against Adversarial Attacks

Generally, the techniques for detecting and defending against adversarial attacks on DNN models can be categorized as reactive and proactive. In a proactive approach, the robustness of the target model is improved via adversarial training [61,62], while, on the other hand, reactive methods aim to detect adversarial attacks by checking the properties of the input data [63]. Several researchers proposed proactive and reactive approaches for detecting and defending against adversarial attacks; for example, [2,31] presented adversarial training methods to defend against adversarial attacks proactively. Another proactive approach is called defensive distillation presented, by [64,65], in which the new model is trained by distilling the information of the hidden layers from the previous model. However, adversarial training and defensive distilling methods increase the training time and resource consumption, which is the main problem in the domain of ADSs. Other methods for proactive defense are model ensembling [66], network regularization [67], and certified robustness [68], while many researchers have also explored reactive defense techniques. This technique can be categorized as adversarial detection and adversarial transformation. The adversarial attack detector detects the adversaries or verifies the feature representation of the input image. It effectively detects poisoning and perturbation attacks where the detector detects manipulated images. Zheng et al. [69] presented an adversarial detector by modeling the intrinsic properties of DNNs. Lee et al. [70] also proposed a framework for detecting adversarial examples that can be used to detect adversaries in any pre-trained Softmax neural classifier. The authors of [71] presented model-pruning-based adversarial attack detection methods, while, on the other hand, Chen et al. [72] presented a method to detect adversarial attacks based on activation clustering methods. In contrast to adversarial detection, the adversarial transformation methods apply transformation techniques to mitigate the effect of adversarial attacks and obtain a clean image. For example, this technique was explored by Jin et al. [73] and proposed a method to eliminate perturbations by means of generative adversarial neural networks. However, the problem with adversarial transformation is that it reduces the efficiency of the DNN model under normal conditions. Therefore, considering the effectiveness of adversarial detectors among the other techniques in detecting adversarial attacks, our paper also proposes a novel deep autoencoder-based adversarial attack detector. In contrast to existing methods, where each method detects only specific types of adversarial attack, our approach can detect any input-image-related attacks such as perturbation, poisoning, and evasion attacks.

The demerits of the existing adversarial attacking model methods can be summarized as follows: First, existing adversarial attacking methods are mostly designed for classification tasks. Second, they are designed to work in an offline setup. Most attacking methods require training loss to perturb input images. However, training loss cannot be accessed during runtime attacks. Adversarial attacks on end-to-end ADSs using existing methods are time-consuming and resource-demanding. Because most methods are query-based iterative, they require high computational resources and time. The demerits of the existing adversarial attack detection and defense methods can be summarized as follows: Most commonly used adversarial defensive methods, such as adversarial training defensive distillation, increase training time and resource consumption; therefore, these methods are hard to implement in the ADS domain. Existing adversarial attack detectors use adversarial transformation techniques to detect adversarial attacks; however, transformation-based methods are unsuitable in the ADS domain as transforming input images reduces the model's efficiency under normal conditions. In contrast to the existing approach, ours does not require high resources, computational time, and transformations. It detects any perturbation-based adversarial attacks.

3. Proposed Method

3.1. Attack Model

The perturbation attack aims to make imperceptible changes in the pixels of input images so that the model becomes dysfunctional. There are two main approaches to gener-

ating adversarial perturbation. The first is a gradient-based method based on FGSM [2] that solves the following equation, Equation (1):

$$\delta^{per} = I + \varepsilon \cdot sign\left(\nabla J_\theta\left(I, l^{true}\right)\right) \tag{1}$$

where ε controls the intensity of the perturbations, ∇J_θ is the cost function, and I is the original image with its true label l^{true}. The second approach is the optimization approach, in which the perturbations are generated by solving the following optimization problem:

$$\delta^{per} = argmin_{I'} \cdot \alpha\left|\left|I - I'\right|\right|_p + l\left(J_{\theta,b'}\left(I'\right)\right) \tag{2}$$

where $argmin_{I'}\ \alpha||I - I'||_p$ calculates the distance between the original image I and the adversarial image I', and $l\left(J_{\theta,b'}\left(I'\right)\right)$ represents the constraints on the loss of the perturbed image I'. Using this equation, we can generate the adversarial image I' similar to the original image I; conversely, the model classifies I' as b', where b' represents the incorrect label for I'.

However, both methods require prior knowledge of the target model. Additionally, obtaining full knowledge of the target model used in autonomous vehicles is very challenging. Another problem with the baseline approach is that these methods were originally designed for the model used in classification tasks. Therefore, we consider our own methods to generate adversarial perturbation that generates perturbation without prior knowledge of the target model in a black-box setting. We considered four naturally occurring noises (i.e., shots, Gaussian, impulse, and speckle noise) and well-known black-box adversarial attacks such as square, HopSkipJump, and decision-based/boundary attacks to perturb input images due to their stealthy nature. Each perturbation can be obtained by solving the following equations.

Let us consider the fact that we have a sequence of original input images $\{I_1, I_2, I_3 \ldots,\ I_t\}$, σ represents the noise parameter, and the noise vector ε, a sequence of perturbed images $\{I_{n,1}, I_{n,2}, I_{n,3} \ldots,\ I_{n,t}\}$, can be generated by using the following equation, Equation (3):

$$I_{adv,\ t} = I_t + \delta_t + \sigma \cdot I_t \cdot \varepsilon_t \tag{3}$$

where t is the time index and δ_t represents the noise vector which is optimized to generate the noise in the image as an adversarial perturbation to maximize the loss of the model at the t^{th} frame. In order to generate a noise vector δ_t, we solve the following equation, Equation (4), with the objective of maximizing the loss of the model for the t^{th} frame:

$$maximize\ L(f(I_t + \delta_t), y_t)\ s.t\ ||\delta_t||_p\ \leq\ \varepsilon \tag{4}$$

where L is the loss function used in the regression model f, and p is a non-negative real number denoting the norm for the noise vector. The noise parameter σ is added to the original input image for each type of perturbation (i.e., shots, Gaussian, impulse, speckle noised, square, HopSkipJump, and decision-based/boundary-attacked image frames). Each perturbation produces its sequence of adversarial images to deceive the target model. For example, when we apply speckle noise perturbation, we obtain a sequence of adversarial images $\{I_{sn,1}, I_{sn,2}, I_{sn,3} \ldots,\ Is_{n,t}\}$ that contains the sn noise, which represents speckle noise. Similarly, when we apply Gaussian noise, we obtain an adversarial image sequence $\{I_{gn,1}, I_{gn,2}, I_{gn,3} \ldots,\ I_{gn,t}\}$ containing Gaussian noise, while $\{I_{stn,1}, I_{stn,2}, I_{stn,3} \ldots,\ I_{stn,t}\}$ represents the sequence for adversarial images when we apply shot noise to perturb the original image sequence. Finally, $\{I_{pn,1}, I_{pn,2}, I_{pn,3} \ldots,\ I_{pn,t}\}$ represents the adversarial image sequences generated when applying impulse noise. The same steps are followed for square, HopSkipJump, and decision-based/boundary attacks, and adversarially perturbed image sequences are generated. Note that the parameter σ is added to each frame individually, while the noise vector δ_t is optimized for each frame independently. The perturbation-based attack is summarized in Algorithm 1. The proposed

algorithm is broken down into three major steps: In step 1, we set various parameters, such as attack intensity ε and noise parameter σ, and δt represents the noise vector. We perturbed the input image in the second step to produce different adversarial image frames by solving the equation mentioned earlier. In the third step, adversarial image frames were fed to the target model to analyze its robustness.

Algorithm 1. An Algorithm for the perturbation-based adversarial attack on the input image I of model $y = f(\theta, I)$ at time stamp t to produce adversarial images $I_{adv,t}$.

Input: The set of input images $\{I_t\}$ where the I_t is the input image at time step t
Output: Adversarial image $I_{adv.t}$ and incorrect trajectory y'.
Parameters: Attack intensity ε, noise parameter σ, and δ_t represents the noise vector
for each time step t **do**
    Perturbation: $I_{adv,t} = I_t + \delta_t + \sigma \cdot I_t \cdot \varepsilon_t$
    Inference : $y' = f(\theta, I_{adv,t})$
end for

3.2. Reconstruction-Based Adversarial Attack Detector

Although various researchers have worked on detecting adversarial attacks, these methods work well for detecting adversaries in classification models. Thus, the applicability of such methods in the regression model for ADSs is unsuitable. For example, defending against and detecting adversarial attacks using the transformation method works based on clipping and rotating input images. However, this can be worked with a classification model to defend rotated and clipped images. Conversely, rotating the input image (e.g., images coming from the front camera) drastically increases the prediction error in the regression setting. Other methods, such as denoising-based adversarial attack defense methods, also apply transformation, and are thus unsuitable for detecting adversaries in the end-to-end model. We present the novel deep-autoencoder-based adversarial attack detector that effectively detects adversaries in the input images. Using our proposed adversarial attack detector, we aim to investigate the impact of perturbation attacks on the robustness of vision-based ADSs.

In order to detect adversarial attacks on vision-based ADSs, we aim to train a deep autoencoder to be used as an adversarial attack detector. An autoencoder is a special type of DNN model that is designed to reconstruct the original input image. A simple autoencoder consists of three components: the encoder, latent presentation, and decoder. The encoder part of the autoencoder maps the given input image I to its low-dimensional representation z with a function $f(I) = z$, and decoder $g(z) = I'$ reconstructs the input image from its low-dimensional representation z. I' represents the reconstructed version of the input image I. During the reconstruction, autoencoders minimize the reconstruction loss $L(I, g(f(x)))$. The function $L(I, g(f(x)))$ measures the difference between the input image and its reconstructed output, and *MSE* is widely used to measure the reconstruction loss. Various kinds of auto-encoders have been proposed, such as convolution autoencoders, variational autoencoders, long short-term memory (LSTM) autoencoders, and deep autoencoders. The number of nodes in each autoencoder's input and output layers must be the same. An autoencoder is considered a deep autoencoder when multiple hidden layers are used in its architecture.

The adversarial attack detector was first trained with the image collected in a normal scenario where perturbations were not applied. Consider that we have a normal training set, $X_{real} = \{I_1, I_2, I_3 \ldots, I_t\}$, and its reconstructed sequence of images from the adversarial attack detector represented by $X_{rec} = \{I'_1, I'_2, I'_3 \ldots, I'_t\}$. The reconstruction loss is denoted by e which is defined by Equation (5) as follows:

$$e_i = d(I_i, I'_i) \tag{5}$$

where d is the Euclidean distance between the original and reconstructed image. Thus, we obtain a set of reconstruction losses $\{e_1, e_2, e_3 \ldots . e_t\}$. By taking the pixel-wise mean squared error, we can calculate $e_i = d\left(I_i, I_i'\right)$ using Equation (6):

$$MSE = \frac{1}{N \cdot H \cdot W \cdot C} \sum_{i=1,j=1,k=1,c=1}^{N,H,W,C} (I[i,j,k,c] - I_r(i,j,k,c))^2 \tag{6}$$

where $N, H, W,\ C$ represent the number of samples, height, width, and color channel, respectively. This reconstruction loss is then compared with the set threshold θ to flag whether the input image is perturbed or normal. Ideally, the reconstruction loss will be less than the set threshold when the input image is normal. On the other hand, when the input image is adversarially perturbed either due to natural distortion or due to a perturbation made by the attacker, the reconstruction loss will be higher. Thus, the detector flags such input streams as adversarial examples. We used a deep autoencoder with nine layers and a combination of different activation functions. Table 1 describes the details of the deep autoencoder used in our approach.

Table 1. Structure of deep autoencoder for adversarial attack detection.

Layer No.	Layer	Type	Activation
Layer_1	Input	Encoder	ReLU
Layer_2	Hidden	Encoder	ReLU
Layer_3	Hidden	Encoder	ReLU
Layer_4	Hidden	Encoder	ReLU
Layer_5	Hidden	Encoder/Decoder	Sigmoid
Layer_6	Hidden	Decoder	ReLU
Layer_7	Hidden	Decoder	ReLU
Layer_8	Hidden	Decoder	ReLU
Layer_9	Output	Decoder	Sigmoid

3.3. Modeling of Framework

The proposed framework for detecting adversarial attacks and the data flow for vision-based ADSs is depicted in Figure 1. In normal cases, the model in vision-based ADSs takes an input image from the front camera and directly maps this input image into the control parameters. The adversarial attack detector produces a reconstruction loss lower than the set threshold for the normal input image, while, on the other hand, when the input stream is perturbed with noise such as shots, Gaussian, impulse, speckle noise, or square, HopSkipJump, and decision-based/boundary attack, the adversarial detector produces a higher reconstruction than the set threshold; thus, this kind of image streams are detected as adversarially attacked image. The reconstruction loss is then fed to the evaluation module, which works in an end-to-end fashion that produces scores for various metrics to evaluate the target model's robustness. Let us consider that we have a regression model $f(\theta, I)$ where θ represents the model parameter of f under normal conditions for the original input image I. The benign output of the model under normal conditions can be defined as

$$y = f(\theta, I) \tag{7}$$

However, our aim is to evaluate the performance of model $f(\theta, I')$ under adversarially attacked image frames I' and to detect those image frames. Therefore, we can define the output of the model $f(\theta, I')$ for the adversarially attacked image I' as follows:

$$y' = f\left(\theta, I'\right) \tag{8}$$

where y' is the adversarial output for the input image I'. Given the input image I, the aim of attacking the model $f(\theta, I)$ is to add perturbation δ^{per} by applying shot, Gaussian, impulse, speckle noise, square, HopSkipJump, or decision-based/boundary attack so that $y' \neq \vec{y}$,

where $\overrightarrow{y}$ is the ground truth driving parameter. When the perturbation δ^{per} is added to the input image I, it must cause a larger deviation between y' and $\overrightarrow{y}$. To quantify the attack intensity, we use the L_2 norm, and to ensure that the attack is imperceptible to human eyes, we bound the attack intensity using the following equation, Equation (9):

$$\left|\left|I' - I\right|\right|_2 = ||\ \delta^{per}||_2\ \leq\ \varepsilon \tag{9}$$

The adversarial attack detector reconstructs each image frame. Based on the reconstruction loss and comparing them to the threshold, the attack detector detects whether or not the input image is adversarial. Furthermore, the end-to-end evaluation framework takes the reconstruction loss *MSE* and deviation in the final output as a score for the proposed evaluation metrics. In this study, our target is to evaluate the robustness of the NVIDIA-DAVE-2 ADS model. The input shape of the target ADS model is $160 \times 320 \times 3$, and the output ranges between $[-1, 1]$. The minus values indicate the ADS is steering toward the left, and positive values indicate it is steering toward the right.

The process for detecting adversarial attacks on the vision-based ADS model is summarized in Algorithm 2. The proposed algorithm to detect the adversarial attacks first takes normal, as well as adversarially perturbed, input images and reconstructs them by minimizing the reconstruction loss. For each image frame, the reconstruction loss is calculated at step 3 using Equation (6), which calculates the pixel-wise reconstruction loss in *MSE*. In step 4, the algorithm compares the reconstruction loss with the threshold and determines whether the image frames are adversarial or normal.

Algorithm 2. Deep-autoencoder-based adversarial attack detection system φ represents the encoder and φ' is the decoder of the trained autoencoder. *MSE* is the mean squared error to calculate the reconstruction loss.

Input: the input images $\{I_t\}$ where I_t is the input image at time step t, threshold θ, and adversarial image frames I_i' where $i = 1, \ldots, N$
Output: Detected adversarial image $I_{adv.t}$, $MSE\ ||I - I'||$

1 $\boldsymbol{\varphi}, \boldsymbol{\varphi}' \leftarrow$ Train the deep autoencoder with the normal image I

2 **for** $i = 1$ **to** N **do**
3 $MSE_i = \left|\left|I_i - g_{\boldsymbol{\varphi}}\left(f_{\boldsymbol{\varphi}'}(I_i)\right)\right|\right|$
4 **If** $MSE_i > \theta$ **then**
5 I_i is an adversarial image.
6 **else**
7 I_i is a normal image.
8 **end if**
9 **end for**

3.4. Evaluation Metrics

To evaluate the robustness of the target model under attack in a quantitative manner at runtime in an end-to-end fashion, we propose a set of evaluation metrics in addition to the offline metrics (TPR, FPR, precision, and F1 score). Each proposed evaluation metric is defined as follows:

(1) *Attack Success Rate (ASR):* Consider that we have x_{trj} representing the trajectory produced for each image frame coming from the front camera. The ASR is defined as $ASR = x'_{trj} - x_{trj} \neq 0$, where x'_{trj} represents the trajectory produced for adversarial image frames. An attack with the intensity of $\varepsilon < 0.003$ can be considered as successful if it successfully deviates the model output from its original value. This means that, if the difference between $x'_{trj} - x_{trj} \neq 0$, the attack has successfully deviated the trajectory either right or left. If the deviation value is negative, the ADS deviates toward the left from its original trajectory, while if it is positive, the ADS deviates toward the right.

(2) *True Detection Rate (TDR):* We defined the TDR as the rate of successfully detecting adversarial image frames by the adversarial attack detector. We set the value of TDR based on the reconstruction loss compared with the set threshold (i.e., 0.035 at runtime). It is the rate of successful detection of adversarial images $I_t{}' = \{I'_1, I'_2, I'_3 \ldots, I'_t\}$ among the normal image frames in the dataset k_{dts}. It can be defined as $TDR = I_t{}'/k_{dts}$. If the adversarial attack detector successfully detects the adversarial image as an adversarial example, the value of TDR is increased by one. In contrast, if it does not flag the adversarial image frames, the TDR value decreases by one.

(3) *False Detection Rate (FDR):* It is the ratio of incorrect detection of adversarial images by the adversarial attack detector. The FDR value is determined by the reconstruction loss, which is calculated using Equation (6), as well as the deviation in the model's output. For any I', if the MSE is lower than the threshold θ, the adversarial attack detector will incorrectly flag the adversarial image as normal. On the hand, the model DNN model's output (i.e., steering angle) will have deviated. Thus, it will create a driving hazard. Therefore, we can determine the value of FDR under two conditions; the adversarial attack detector incorrectly detects the adversarial image as normal and the $\Delta x_{trg} = x'_{trj} - x_{trj}$ must be greater than 0, where Δx_{trg} represents the deviation in vehicle trajectories.

(4) *Normal Driving Rate (NDR):* The NDR is defined as the rate of the normal driving trajectory from its total trajectory x_{trj} where there is no deviation in steering angle observed under adversarial attacks. Note that, in this study, the DNN model was trained for a lane-following purpose. Therefore, the NDR is determined when the ADSs follow the center of the road without any deviation in the actual steering angle under adversarial attacks. Therefore, NDR is defined as $NDR = x'_{trj} - x_{trj} \approx 0$ under adversarial attacks.

4. Experimental Results

4.1. Training the Autonomous Driving Model

We use the Udacity [74] self-driving car simulator in this experiment. We chose the simulation platform to test the robustness of the target model since it is hazardous for driving safety to test on real systems [75–77]. The Udacity simulator provides options to collect driving scenes to train the model for behavioral cloning and testing the trained model in autonomous driving mode. We collected a 34 K image dataset to train the model for lane-keeping purposes. The deep-autoencoder-based adversarial attack detector was validated using an additionally collected dataset of 19 K images with different added noises (2 K impulse-, 2 K Gaussian-, 3 K shot-, and 3 K speckle-noised adversarially attacked images). We also collected 3 K square attacks, 3 K HopSkipJump, and 3 K decision-based adversarially attacked images.

These images were collected to validate the attack detector on offline metrics such as TPR, FPR, precision, and F1 score. We then trained the NVIDIA-DAVE-2 model and proposed the deep-autoencoder-based adversarial attack detector with the dataset under normal conditions. After training the ADS model and the adversarial detector, we applied the adversarial attacks at run time and evaluated the robustness of the target model. During this experiment, we trained models for 500 epochs on a system equipped with core i9 processors and 64 GB memory. The system was also equipped with an Nvidia GeForce RTX 3090 GPU.

4.2. Online Adversarial Attack Detection

Normally, when adversarial attacks are applied to classification tasks in offline settings, the perturbation is applied to the image to maximize the training loss. For example, the Goodfellow et al. [2] approach maximized the training loss and used the gradient of the training loss to generate the perturbation. However, runtime attacks cannot access training

loss; consequently, we cannot calculate the loss. Therefore, existing attacking methods cannot effectively be applied to attack the regression model at runtime.

We can either increase or decrease the model output by attacking the model at runtime. In the end-to-end ADS context, we can deviate the ADSs towards the left by decreasing the model output or deviate right by increasing the model output. As mentioned earlier, we cannot access training loss at runtime. Therefore, we use our own method to add different types of naturally occurring noise to increase or decrease the output by using the equations defined earlier:

$$I_{adv,\ t} = I_t + \delta_t + \sigma \cdot I_t \cdot \varepsilon_t \tag{10}$$

The δ_t factor can be achieved by optimizing the following equation:

$$maximize\ L(f(I_t + \delta_t), y_t)\ s.t\ ||\delta_t||_p\ \leq\ \varepsilon \tag{11}$$

where L is the loss function for model f, and p is the norm for the noise vector. The noise parameter σ is added to the original input image as in the expression for each type of perturbation. Hence, an attacker can force the vehicle to the left side by decreasing the model output by generating perturbations such as shot, impulse noise, etc. The adversarial attack detector then reconstructs each frame and compares the reconstruction loss to the automatically set threshold. Figure 2 shows the sample image frame with adversarial perturbations using different noises and their respective reconstruction loss in *MSE*.

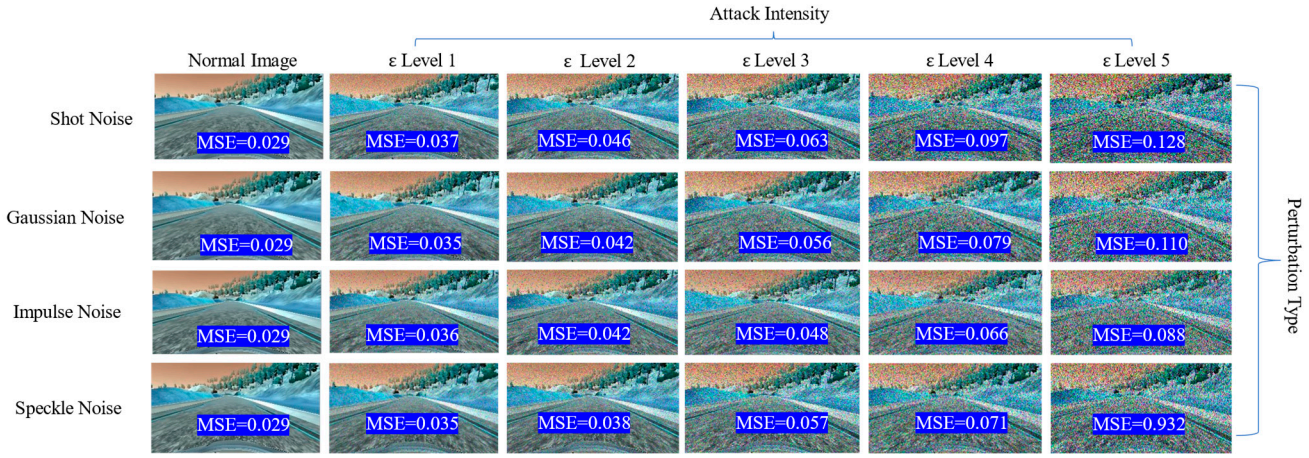

Figure 2. The adversarial attack intensity and level of distortion in image quality. As the ε increases, the image distortion also increases; thus, the adversarial attack detector flags these images as adversarial images. The *MSE* is the reconstruction loss of the deep autoencoder, and it is compared with the threshold $\theta = 0.035$. Sample images show the road scene in the Udacity simulator.

4.3. Evaluation

To evaluate the performance of the adversarial attack detector, we use the *TPR*, *FPR*, *F*1 score, and precision metrics. We also use the proposed metrics ASR, TDR, FDR, and NDR to evaluate the model at runtime. The *TPR* and *FPR* can be calculated by using Equations (12) and (13), respectively:

$$TPR = \frac{TP}{TP + FN} \tag{12}$$

$$FPR = \frac{FP}{TN + FP} \tag{13}$$

The adversarial attack detector aims to increase the *TPR* while decreasing the *FPR*. To evaluate the effectiveness of the proposed approach in detecting adversarial attacks, we present our experimental results in graphical, as well as tabular, form. Figure 2 shows

that, as the attack intensity increases, the reconstruction loss for each frame also increases. Compared to the reconstruction loss of the normal images, which is 0.29, the reconstruction loss for adversarial image frames increases more as the attack intensity increases. Thus, the adversarial attack detector effectively detects by comparing this reconstruction loss against the set threshold of 0.035. It can be observed in Figure 2 that all the reconstruction loss is almost higher than the threshold, indicating that the detector successfully detects those adversarial frames at runtime. A higher reconstruction loss indicates a higher degradation in image quality. As a result, this makes it hard for the model to understand the road features and causes it to start deviating from its original trajectory, resulting in no longer following the lane-keeping requirement. We can clearly observe in Figure 2 that, when the level of ε is set to 1, the difference between the original and adversarial image is very low—however, the reconstruction loss was recorded to be around 0.035.

Interestingly, the adversarial attack detector still detects those minor levels of perturbation, indicating the proposed approach's effectiveness. However, as the level of ε increases, the difference between the original and adversarial image is clearly visible. In some cases, for example, when the level of ε was set to 5, the image quality drastically decreased, and the reconstruction loss drastically increased. For example, we can observe in Figure 2 that the highest reconstruction loss was 0.932 when we applied level 5 speckle noise perturbation. The lowest reconstruction loss was recorded when we applied level 1 Gaussian and speckle noise, respectively, indicating that the detector is effective even if we applied a minor level of perturbation. However, at offline evaluation using the TPR, FPR, F1 score, and precision metrics, we set the threshold at 0.05 because there are very few adversarial images compared to normal images. Therefore, lowering the threshold below 0.05 makes the detection results almost useless, as both TPR and FPR become zero.

Based on the extensive experimentation, we set the threshold to distinguish between the adversarial input vs. normal input. While setting the threshold at offline evaluation, we found that $\theta = 0.05$ achieved better detection with allowable balanced false positive rates. The threshold below 0.05 causes both TPR and FPR to be zero, causing the adversarial detector performance to be useless. A threshold greater than 0.05 also decreases the TPR compared to FPR. Hence, we set 0.05 as the threshold for offline evaluation and 0.035 for online evaluation.

We present our quantitative results in Table 2 using the TPR, FPR, F1 score, and precision metrics. Our proposed detector effectively achieved a higher TPR and lower FPR, as well as a good F1 score and precision in detecting adversarial attacks. Table 2 shows that the highest TPR, which is 93%, was achieved when the perturbation type was set to speckle noise and shot noise. We also observed a very low FPR of 9% and 9.6% for speckle- and shot-noised perturbation, while, in the case of Gaussian-noise-based perturbation, the adversarial attack detector could achieve 83% of TPR vs. 14.6% FPR. The same phenomenon was observed while detecting impulse-noise-based adversarial attacks. The detector could achieve 82% TPR vs. 10% FPR. The difference between the detection rates was unexpected. However, during our experiment, we found two reasons for this phenomenon: (1) The nature of these perturbations is different from each other; consequently, the performance of the deep autoencoder also varies. (2). We found that the number of image frames in the dataset also impacted the detection results. The number of images containing Gaussian and impulse noise is smaller than the speckle and shot noise. The performance of the deep autoencoder also achieved excellent results on both the F1 score and precision metrics. We can see that the highest F1 score, which is 51.6%, was achieved while detecting the Gaussian-noise-based adversarial attacks. To attack the vision-based ADSs at runtime with different intensities, we adjust the noise term σ and the attack intensity ε in (10).

Table 2. Performance of adversarial attack detector in detecting various perturbation-based adversarial attacks.

θ = 0.05				
Perturbation Types	**TPR**	**FPR**	**F1 Score**	**Precision**
Shot Noise	93%	9.4%	41%	26.2%
Gaussian Noise	83%	14.6%	51.6%	37.3%
Impulse Noise	82%	10%	33%	20.7%
Speckle Noise	93%	9%	41%	26.2%
Square	93.1%	9.4%	40.2%	26%
HopSkipJump	93%	9%	40.8%	26%
Decision-based/Boundary attacks	93.5%	9.2%	41.5%	27%

We set five levels of attack intensities to perturb the images in either a multiplicative way or an additive way. Level 1 represents the lowest perturbation level in the original image's pixel, and level 5 represents the highest. During our experiment, we found that, even with a minor level of perturbation, we achieved the highest attack success rate. Figure 3 presents the impact of the adversarial attack on the NVIDIA-DAVE-2 ADS model. In all types of adversarial attacks, we achieved 100% ASR, and all attacks caused strong deviations in the original trajectory of the ADSs. We achieved 0% NDR in all types of adversarial attacks. This means that all adversarial attacks cause significant violations of lane-keeping requirements. The impact of perturbation attacks on the target model is presented in Figures 4 and 5. We can see that the reconstruction loss also increased with the increased level of attack intensity. Consequently, the deviation in the actual steering angle is substantially increased. Let us take the example of Figure 4a,b, representing the reconstruction loss of the deep autoencoder and the actual deviation in steering angle when the shot-noise-induced adversarial attack was applied. These figures show that the shot-noised adversarial attack significantly deviates the steering angle towards the left. As discussed earlier, the NVIDIA-DAVE ADS model's final output ranges between $[-1, 1]$. Therefore, when the steering angle has a negative value, it indicates it is driving toward the left, while a positive steering angle indicates ADS is steering toward the right. Therefore, we can observe that, when we applied the shot noise adversarial attack, the ADS was forced to drive toward the left. We observed a -0.5288 mean deviation in the steering angle when only the lowest shot-noised adversarial perturbation (i.e., ε level 1) was applied. As the attack intensity increased, the reconstruction loss and deviation in steering angle increased.

For example, when we applied the highest level (i.e., ε level 5) of Gaussian noise adversarial attack, the reconstruction loss drastically increased to 0.08. We can observe in Figure 4d that there is a significant deviation in the steering angle. The mean deviation in steering angle due to a level 5 Gaussian-noised adversarial attack was observed to go up to -0.6006. The same trends were observed in the case of impulse and speckle noise adversarial attacks, as can be seen in Figure 5a, b, c, and d, respectively. We can observe that all deviations were observed towards the left as the deviation contains negative values. Hence, when we applied all types of adversarial attacks in an untargeted setting without access to the model structure nor the final output, all the deviations were recorded toward the left side. The mean deviation for all types of adversarial attacks is presented in Table 3. From Table 3, we can see that all types of adversarial attacks drastically impact the final output. Note that, to detect these attacks at runtime, we set $\theta = 0.035$. Thus, the adversarial attack detector flags all those image frames as adversarial frames whose reconstruction loss is higher than 0.035. Table 4 shows the attack detector could detect almost 100% of level 3, 4, and 5 adversarial attacks. The minimum detection result was 54.5% observed in the case of level 1 Gaussian-noise-based adversarial attacks. The Level 1 Gaussian-noise-based adversarial attack was the only case when the highest FDR (i.e., 45.5%) of adversarially attacked image frames were observed. We found two scenarios to be discussed in the Results sections. The first one is that we reported the highest FDR (i.e., 45.5%.) when the level 1 Gaussian noise adversarial attack was applied, while, on the other hand, in the case

of level 1 shot, impulse, and speckle adversarial noise attacks, the detector could detect 90%, 67.5%, and 89.6% of all adversarial attacks, respectively, with an FDR of 10.3%, 32.6%, and 10% at runtime. We found only a minor drop in the detection rate during the experiment when we applied the level 2 perturbation. The attack detector could detect 96.1% of level 2 speckles, 98.7% of Gaussian, and 97.4% of all shot noise adversarial attacks. To analyze the impact of the adversarial attack in an online fashion, we presented the quantitative experimental results in Table 4.

Figure 3. Impact of adversarial attack on the original trajectory of the target ADS. These samples were collected when the impulse-noise-based adversarial attack was applied. (**a**) represents the initial position, and (**b**,**c**) show the impact of the attack. We can see that the ADS started deviating from its original trajectory. (**d**) shows that ADS is no longer driving on the road and crashed on the roadside. (**e**,**f**) represents the samples of adversarial image frames from front camera of ADS.

Table 3. Mean deviation in steering angle (i.e., the final output of the target model).

ε	Shot Noise	Gaussian Noise	Impulse Noise	Speckle Noise	Square Attacks	HopSkipJump Attacks	Boundary Attacks
Level 1	−0.5288	−0.6055	−0.6119	−0.5966	−0.151650	−0.1629560	−0.149258
Level 2	−0.6241	−0.6126	−0.6051	−0.6194	−0.151652	−0.1629606	−0.149260
Level 3	−0.5793	−0.6241	−0.5624	−0.6054	−0.151654	−0.1629607	−0.1492638
Level 4	−0.5221	−0.6021	−0.5712	−0.6293	−0.151654	−0.1629642	−0.1492673
Level 5	−0.3051	−0.6006	−0.5324	−0.5714	−0.151657	−0.1629667	−0.1492689

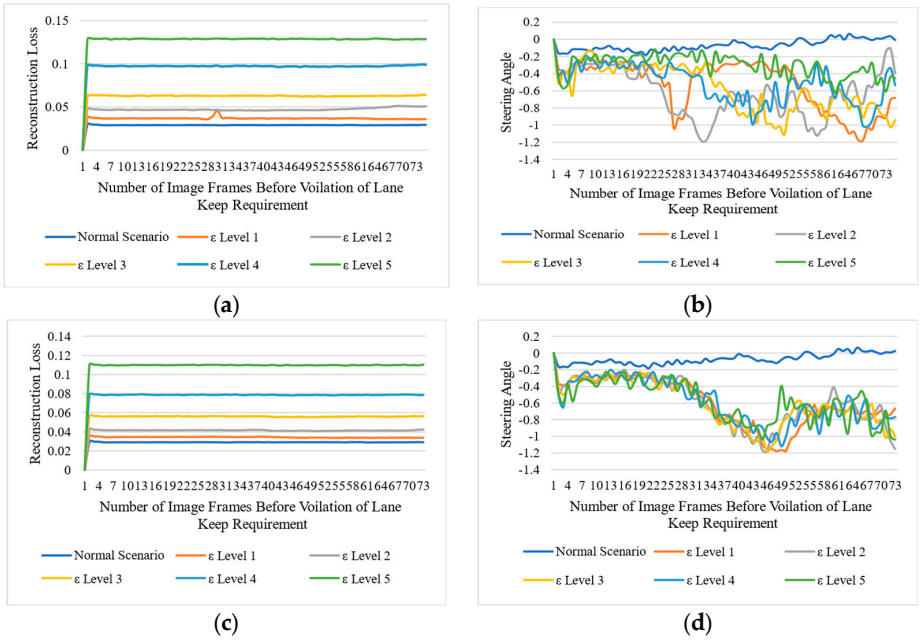

Figure 4. Impact of different types of perturbation-based adversarial attacks on vision-based ADSs. The left column represents the reconstruction loss of the deep autoencoder when the threshold $\theta = 0.035$, and the right column represents the deviations in steering angles. (**a**,**c**) represent the reconstruction losses for shot- and Gaussian-noise-based perturbation attacks, respectively, while (**b**,**d**) represent the actual deviation in steering angle due to shot- and Gaussian-noise-based perturbation attacks, respectively.

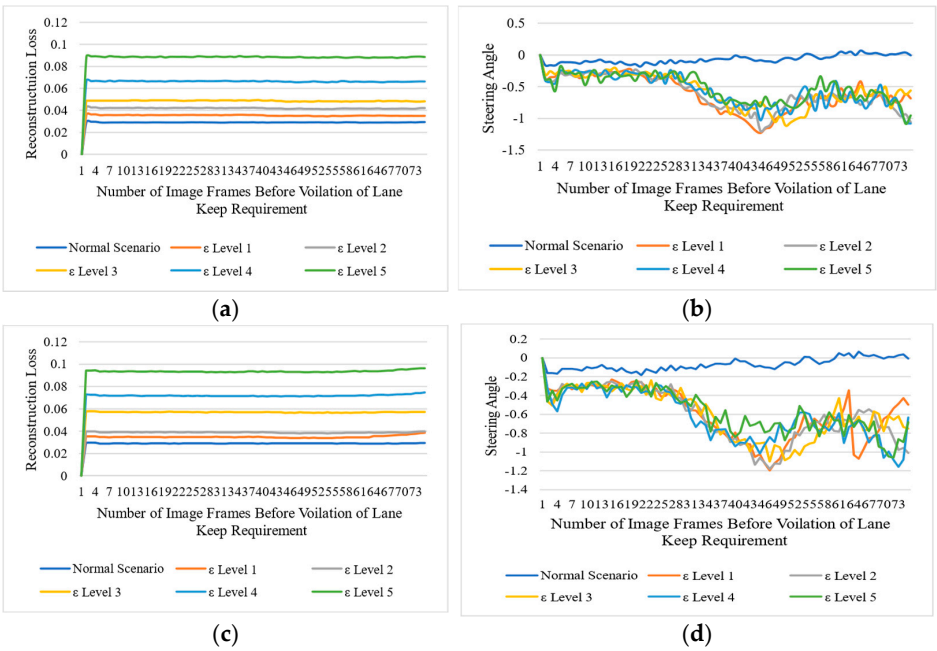

Figure 5. Impact of different types of perturbation-based adversarial attacks on vision-based ADSs. The left column represents the reconstruction loss of the deep autoencoder when the threshold $\theta = 0.035$, and the right column represents the deviations in steering angle. (**a**,**c**) represent the reconstruction losses for impulse- and speckle-noise-based perturbation attacks, respectively, while (**b**,**d**) represent the actual deviation in steering angle due to impulse- and speckle-noise-based perturbation attacks, respectively.

Table 4. Performance of adversarial attack detectors on online evaluation metrics. We achieved 100% ASR, resulting in 0% NDR while evaluating the systems at runtime. Therefore, ASR and NDR metrics were not included in the table.

θ=0.035														
ε	**Shot Noise**		**Gaussian Noise**		**Impulse Noise**		**Speckle Noise**		**Square Attack**		**HopSkipJump Attack**		**Boundary Attack**	
	TDR (%)	**FDR (%)**	**TDR (%)**	**FDR (%)**	**TDR (%)**	**FDR (%)**	**TDR (%)**	**FDR (%)**	**TDR (%)**	**FDR (%)**	**TDR (%)**	**FDR (%)**	**TDR (%)**	**FDR (%)**
Level 1	90	10	54.5	45.5	67.5	32.6	89.6	10.3						
Level 2	97.4	2.5	98.7	1.2	100	0	96.1	3.8						
Level 3	100	0	100	0	100	0	100	0	100	0	100	0	100	0
Level 4	100	0	100	0	100	0	100	0						
Level 5	100	0	100	0	100	0	100	0						

We did not report the ASR and NDR in the table, as the ASR is inversely proportional to the NDR. If the ASR is 100%, then the NDR will automatically be 0%. Because all attacks were successful in deviating the steering angle, the ADSs will no longer follow the actual trajectory. Thus, we achieved 100% ASR during our experiment, as shown in Figures 4 and 5, and Table 4. All attacks caused a significant deviation in steering angle with more than -0.3051 mean deviation. Consequently, we observed 0% NDR under attacks. Therefore, we did not report this in Table 4. On the other hand, we achieved 100% TDR when applying levels 3, 4, and 5 adversarial attacks with no FDR at runtime. Conversely, we also observed significant drops in TDR and increased FDR when we applied the level 1 and 2 adversarial attacks. Taking a level 1 Gaussian-noise-based adversarial attack from Table 4 as an example, we can see that the TDR is 54.5% which is quite low, while FDR is significantly high (i.e., 45.5%). Same as when ADS was attacked with the intensity-level-1 impulse noise, we observed a considerable drop in TDR, which is 67.5%, with an FDR of 32.6%. However, in all other cases, except for the level 2 adversarial attack, the detector could detect all adversaries with 100% TDR. The proposed adversarial attack detectors performed significantly well in detecting other black-box attacks, such as square attacks, HopSkipJump, and decision-based/boundary attacks in terms of the online evaluation framework. Table 4 and Figure 6 depict the quantitative performance evaluation of adversarial attack detectors on online evaluation metrics.

Taking the square attack as an example from Table 4, we can see that the attack detector could detect 100% of square attacks in the online setting. Similarly, other attacks, such as HopSkipJump and decision-based/boundary attacks, were also detected with 100% TDR. Figure 6 shows the impact of square, HopSkipJump, and decision-based boundary attacks and their detection rate when the threshold was set to 0.035. In contrast to shot-, Gaussian-, impulse-, and speckle-noise-based perturbation attacks, all deviations in steering angle were observed toward the right (i.e., positive values indicate ADSs towards the right). Figure 6b,d,f show the deviation in steering angle toward the right. An interesting phenomenon observed in the ADS was the attack with square, HopSkipJump, and decision-based attacks. When the intensity of the attack was increased, there was no significant increase in the reconstruction loss. A minor change in the reconstruction loss was observed when increasing the attack intensities. However, even at a minor level (i.e., $\varepsilon = 0.1$), the reconstruction loss was significantly increased to 0.082, exceeding the set threshold. Hence, all adversarial attacks were successfully detected by the attack detector.

Figure 6. Impact of square, HopSkipJump, and decision-based adversarial attacks on ADSs. The left column represents the reconstruction loss of the deep autoencoder when the threshold $\theta = 0.035$, and the right column represents the deviations in steering angle. (**a**,**c**,**e**) represent the reconstruction losses for square, HopSkipJump, and decision-based/boundary attacks, while (**b**,**d**,**f**) represent the actual deviation in steering angle due to square, HopSkipJump, and decision-based/boundary attacks, respectively.

4.4. Comparative Analysis with Existing Approaches

We compared attacking models, the approach to detect adversarial attacks, and the performance of the proposed approach with [78,79]. We considered the most commonly used black-box attacks to attack the target model because white-box attacks are hard to implement in ADSs in real time. The attacker requires full access to the target model to implement white-box attacks. It is hard to implement white-box attacks on ADSs in the real world. We compare our approach with the existing works in terms of detection accuracy. Our deep-autoencoder-based attack detector achieved a high detection rate in detecting black-box attacks. The detection accuracy of our proposed method was compared with [79] in the black-box setting. Compared to [79], our approach achieved an average of 94% detection rate while detecting black-box-type adversarial attacks. We achieved the highest detection rate of 97.4 while detecting shot-noised adversarial attacks, while the lowest detection rate of 90% was recorded when detecting Gaussian-noise-based adversarial attacks. On the other hand, the highest detection rate of the detector proposed in [79] was recorded at 74.1% in detecting FGSM attacks in the black-box setting.

Our adversarial attack detector also outperformed the feature-squeezing technique proposed by the authors in [78]. We compared the highest accuracies achieved by the feature-squeezing technique versus our technique. The feature-squeezing technique achieved the highest accuracy of 83.88% when detecting Carlini- and Wagner-type attacks. On the other hand, it could detect only 67.47% of the Gaussian noise attack. In contrast, our autoencoder-

based adversarial attack detector could detect up to 90.6% of Gaussian noise attacks, as shown in Table 5.

Table 5. Comparative performance analysis of proposed adversarial attack detector vs. existing approaches.

Reference Methods	Attack Methods		Attack Detection Rate	Detection Method
[79]	FGSM		74.1%	Autoencoder and Memory Module
	AdvGAN		64.4%	
[78]	C&W		83.88%	Feature Squeezing
	Gaussian Noise		67.47%	
	Brightness		66.69%	
This Study	Epsilon Attacks	Shot Noise	**97.4%**	Deep Autoencoder
		Gaussian Noise	**90.6%**	
		Impulse Noise	**93.5%**	
		Speckle Noise	**97.14%**	
	Square Attack		**93.1%**	
	HopSkipJump Attack		**93%**	
	Threshold Attack		**93.5%**	

4.5. Summary of the Findings

The findings from our experiment results can be summarized as follows:

(1) State-of-the-art end-to-end models for vision-based ADS systems also suffer from the lack of robustness against adversarial attacks. Even a minor adversarial attack can significantly deviate the vehicle from its original trajectory.
(2) Autoencoder-based adversarial attack detectors can detect any type of perturbation-based, poisoning, and evasion-type attacks. All image-specific attacks can be detected effectively using autoencoders.
(3) Only evaluating the trained model with a previously collected dataset does not necessarily ensure the model's robustness. For example, we discussed the notion that offline evaluations are data-specific and sensitive to the previously collected data distribution. Therefore, the model should be evaluated in an online fashion to check the robustness of the model in real scenarios at runtime.

4.6. Threats to Validity

The first threat to the validity of our proposed approach is that we implemented the experiment in simulation environments. However, it was necessary to implement the proposed approach in a simulation environment, as validating the proposed system with a real system in a controllable way is impossible. However, authors [80] state that the data generated using a simulator produces the same results as those generated from real-world systems. Therefore, we expect that implementing the proposed method on a real system will yield the same result.

Another threat to the validity of the proposed system is that we use a few types of perturbation adversarial attacks that the attackers can launch or that can occur due to natural phenomena. This imposes a constraint on the generalizability of the proposed approach. However, we tried to mitigate this threat by choosing the state-of-the-art NVIDIA ADS model to validate our approach. We believe that, if a state-of-the-art model like NVIDIA can suffer from the lack of robustness to these perturbation-based adversarial attacks, others may also exhibit the behavior. The last threat to our approach's validity is the selection of deep autoencoders. The performance of the deep autoencoders may vary depending on the autoencoder's architecture. However, we mitigated this threat by

validating the performance of the proposed system using four different perturbation-based adversarial attacks. From the experimental results, we can see that the performance of the attack detector is stable for all types of adversarial attacks.

5. Conclusions

This paper investigated the impact of perturbation-based adversarial attacks on the DNN model in vision-based ADSs. We exploited the perturbation-based adversarial attack's impact on the target model's robustness. We first proposed four different types of naturally occurring noises to create perturbations in order to target attack models that manipulate the pixel values of the original images, and then we presented an adversarial attack detector using a deep autoencoder. We also evaluated the effectiveness of the proposed detector using modified state-of-the-art black-box adversarial attack models such as square, HopSkipJump, and decision-based attacks. Finally, we presented the end-to-end evaluation framework to evaluate the impact of adversarial attacks, as well as the performance of the attack detector at runtime. Through extensive experimental evaluation, we found all types of proposed perturbation attacks cause a significant deviation in the final output of the target model. All the attacks directly impacted the robustness of the target models and forced the ADSs to deviate from the original trajectories. The proposed deep-autoencoder-based adversarial attack detector achieved excellent results in detecting adversarial attacks at runtime. The proposed evaluation framework also helps to evaluate the robustness of the model at runtime, which is hard to evaluate using the existing offline evaluation metrics. Our experimental results provide evidence of the vulnerabilities of models used in safety-critical applications such as ADSs. The proposed approach can help to select a robust model for safety-critical applications and gives new pathways to explore the potential of autoencoders in detecting adversarial attacks.

Based on the experimental results obtained, we can explore the impact of other attacks, such as physical (i.e., patch attacks) attacks, on the DNN models used in ADSs. These attacks pose a significant real-world threat to safety-critical applications in the intelligent transportation domain and require robust countermeasures. Since patch attacks (such as adversarial patches on traffic signs) are more realistic after black-box attacks on ADSs, our forthcoming work will focus on detecting and mitigating patch attacks. Our future work will focus on developing detective and defensive mechanisms by leveraging advanced techniques such as generative adversarial neural networks and denoising autoencoders to effectively mitigate the adversarial perturbations and patches so that the resilience and robustness of the ADS model against patch attacks can be enhanced.

Author Contributions: Conceptualization, M.H. and J.-E.H.; methodology, M.H.; software, M.H.; validation, M.H. and J.-E.H.; formal analysis, M.H.; investigation, M.H.; resources, J.-E.H.; data curation, M.H. and J.-E.H.; writing—original draft preparation, M.H.; writing—review and editing, J.-E.H.; visualization, M.H.; supervision, J.-E.H.; project administration, J.-E.H.; funding acquisition, J.-E.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (RS-2023-00237203).

Data Availability Statement: The data presented in this study can be made available upon request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to End Learning for Self-Driving Cars. *arXiv* **2016**, arXiv:1604.07316.
2. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
3. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing Properties of Neural Networks. In Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014.

4. Li, Y.; Cheng, M.; Hsieh, C.J.; Lee, T.C.M. A Review of Adversarial Attack and Defense for Classification Methods. *Am. Stat.* **2022**, *76*, 329–345. [CrossRef]
5. Liu, P.; Fu, H.; Ma, H. An End-to-End Convolutional Network for Joint Detecting and Denoising Adversarial Perturbations in Vehicle Classification. *Comput. Vis. Media* **2021**, *7*, 217–227. [CrossRef]
6. Vemparala, M.R.; Frickenstein, A.; Fasfous, N.; Frickenstein, L.; Zhao, Q.; Kuhn, S.; Ehrhardt, D.; Wu, Y.; Unger, C.; Nagaraja, N.S.; et al. BreakingBED: Breaking Binary and Efficient Deep Neural Networks by Adversarial Attacks. *Lect. Notes Netw. Syst.* **2022**, *294*, 148–167. [CrossRef]
7. Gurina, A.; Eliseev, V. Quality Criteria and Method of Synthesis for Adversarial Attack-Resistant Classifiers. *Mach. Learn. Knowl. Extr.* **2022**, *4*, 519–541. [CrossRef]
8. Pereira, A.; Thomas, C. Challenges of Machine Learning Applied to Safety-Critical Cyber-Physical Systems. *Mach. Learn. Knowl. Extr.* **2020**, *2*, 579–602. [CrossRef]
9. Bendiab, G.; Hameurlaine, A.; Germanos, G.; Kolokotronis, N.; Shiaeles, S. Autonomous Vehicles Security: Challenges and Solutions Using Blockchain and Artificial Intelligence. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 3614–3637. [CrossRef]
10. Puttagunta, M.K.; Ravi, S.; Nelson Kennedy Babu, C. Adversarial Examples: Attacks and Defences on Medical Deep Learning Systems. *Multimed. Tools Appl.* **2023**, *82*, 33773–33809. [CrossRef]
11. Ling, X.; Wu, L.; Zhang, J.; Qu, Z.; Deng, W.; Chen, X.; Qian, Y.; Wu, C.; Ji, S.; Luo, T.; et al. Adversarial Attacks against Windows PE Malware Detection: A Survey of the State-of-The-Art. *Comput. Secur.* **2023**, *128*. [CrossRef]
12. Schwinn, L.; Raab, R.; Nguyen, A.; Zanca, D.; Eskofier, B. Exploring Misclassifications of Robust Neural Networks to Enhance Adversarial Attacks. *Appl. Intell.* **2023**, *2021*, 103134. [CrossRef]
13. Zhang, J.; Chen, L.; Liu, B.; Ouyang, B.; Xie, Q.; Zhu, J.; Li, W.; Meng, Y. 3D Adversarial Attacks beyond Point Cloud. *Inf. Sci.* **2023**, *633*, 491–503. [CrossRef]
14. Sadrizadeh, S.; Dolamic, L.; Frossard, P. TransFool: An Adversarial Attack against Neural Machine Translation Models. *arXiv* **2023**, arXiv:2302.00944.
15. Zhang, F.; Christakis, M. DeepSearch: A Simple and Effective Blackbox Attack for Deep Neural Networks. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, 8–13 November 2020. [CrossRef]
16. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
17. Modas, A.; Sanchez-Matilla, R.; Frossard, P.; Cavallaro, A. Toward Robust Sensing for Autonomous Vehicles: An Adversarial Perspective. *IEEE Signal Process. Mag.* **2020**, *37*, 14–23. [CrossRef]
18. Deng, Y.; Zheng, X.; Zhang, T.; Chen, C.; Lou, G.; Kim, M. An Analysis of Adversarial Attacks and Defenses on Autonomous Driving Models. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications (PerCom), Austin, TX, USA, 23–27 March 2020.
19. Li, Y.; Wen, C.; Juefei-Xu, F.; Feng, C. Fooling LiDAR Perception via Adversarial Trajectory Perturbation. In Proceedings of the IEEE International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021.
20. Wang, X.; Cai, M.; Sohel, F.; Sang, N.; Chang, Z. Adversarial Point Cloud Perturbations against 3D Object Detection in Autonomous Driving Systems. *Neurocomputing* **2021**, *466*, 27–36. [CrossRef]
21. Zhang, Q.; Hu, S.; Sun, J.; Alfred Chen, Q.; Morley Mao, Z. On Adversarial Robustness of Trajectory Prediction for Autonomous Vehicles. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022.
22. Li, Y.; Xu, X.; Xiao, J.; Li, S.; Shen, H.T. Adaptive Square Attack: Fooling Autonomous Cars with Adversarial Traffic Signs. *IEEE Internet Things J.* **2021**, *8*, 6337–6347. [CrossRef]
23. Tu, J.; Ren, M.; Manivasagam, S.; Liang, M.; Yang, B.; Du, R.; Cheng, F.; Urtasun, R. Physically Realizable Adversarial Examples for LiDAR Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020.
24. Zhang, J.; Lou, Y.; Wang, J.; Wu, K.; Lu, K.; Jia, X. Evaluating Adversarial Attacks on Driving Safety in Vision-Based Autonomous Vehicles. *IEEE Internet Things J.* **2022**, *9*, 3443–3456. [CrossRef]
25. Kong, Z.; Guo, J.; Li, A.; Liu, C. PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, WA, USA, 16–18 June 2020.
26. Schulze, J.P.; Sperl, P.; Böttinger, K. DA3G: Detecting Adversarial Attacks by Analysing Gradients. In Proceedings of the 26th European Symposium on Research in Computer Security, Darmstadt, Germany, 4–8 October 2021.
27. Hwang, U.; Park, J.; Jang, H.; Yoon, S.; Cho, N.I. PuVAE: A Variational Autoencoder to Purify Adversarial Examples. *IEEE Access* **2019**, *7*, 126582–126593. [CrossRef]
28. Bakhti, Y.; Fezza, S.A.; Hamidouche, W.; Deforges, O. DDSA: A Defense against Adversarial Attacks Using Deep Denoising Sparse Autoencoder. *IEEE Access* **2019**, *7*, 160397–160407. [CrossRef]

29. Liao, F.; Liang, M.; Dong, Y.; Pang, T.; Hu, X.; Zhu, J. Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
30. Saha, S.; Kumar, A.; Sahay, P.; Jose, G.; Kruthiventi, S.; Muralidhara, H. Attack Agnostic Statistical Method for Adversarial Detection. In Proceedings of the 2019 International Conference on Computer Vision Workshop, ICCVW 2019, Seoul, Republic of Korea, 27–28 October 2019.
31. Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; McDaniel, P. Ensemble Adversarial Training: Attacks and Defenses. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
32. Ren, K.; Wang, Q.; Wang, C.; Qin, Z.; Lin, X. The Security of Autonomous Driving: Threats, Defenses, and Future Directions. *Proc. IEEE* **2020**, *108*, 357–372. [CrossRef]
33. Almutairi, S.; Barnawi, A. Securing DNN for Smart Vehicles: An Overview of Adversarial Attacks, Defenses, and Frameworks. *J. Eng. Appl. Sci.* **2023**, *70*, 1–29. [CrossRef]
34. Aung, A.M.; Fadila, Y.; Gondokaryono, R.; Gonzalez, L. Building Robust Deep Neural Networks for Road Sign Detection. *arXiv* **2017**, arXiv:1712.09327.
35. He, F.; Chen, Y.; Chen, R.; Nie, W. Point Cloud Adversarial Perturbation Generation for Adversarial Attacks. *IEEE Access* **2023**, *11*, 2767–2774. [CrossRef]
36. Cao, Y.; Zhou, Y.; Chen, Q.A.; Xiao, C.; Park, W.; Fu, K.; Cyr, B.; Rampazzi, S.; Morley Mao, Z. Adversarial Sensor Attack on LiDAR-Based Perception in Autonomous Driving. In Proceedings of the ACM Conference on Computer and Communications Security, London, UK, 11–15 November 2019.
37. Nassi, D.; Ben-Netanel, R.; Elovici, Y.; Nassi, B. MobilBye: Attacking ADAS with Camera Spoofing. *arXiv* **2019**, arXiv:1906.09765.
38. Chi, L.; Msahli, M.; Memmi, G.; Qiu, H. Public-attention-based Adversarial Attack on Traffic Sign Recognition. In Proceedings of the 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023.
39. Yang, X.; Liu, W.; Zhang, S.; Liu, W.; Tao, D. Targeted Attention Attack on Deep Learning Models in Road Sign Recognition. *IEEE Internet Things J.* **2021**, *8*, 4980–4990. [CrossRef]
40. Patel, N.; Krishnamurthy, P.; Garg, S.; Khorrami, F. Overriding Autonomous Driving Systems Using Adaptive Adversarial Billboards. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 11386–11396. [CrossRef]
41. Ghosh, A.; Mullick, S.S.; Datta, S.; Das, S.; Das, A.K.; Mallipeddi, R. A Black-Box Adversarial Attack Strategy with Adjustable Sparsity and Generalizability for Deep Image Classifiers. *Pattern Recognit.* **2022**, *122*, 108279. [CrossRef]
42. Choi, J.I.; Tian, Q. Adversarial Attack and Defense of YOLO Detectors in Autonomous Driving Scenarios. In Proceedings of the IEEE Intelligent Vehicles Symposium, Aachen, Germany, 5–9 June 2022.
43. Jia, W.; Lu, Z.; Zhang, H.; Liu, Z.; Wang, J.; Qu, G. Fooling the Eyes of Autonomous Vehicles: Robust Physical Adversarial Examples Against Traffic Sign Recognition Systems. *arXiv* **2022**, arXiv:2201.06192.
44. Jiang, W.; Li, H.; Liu, S.; Luo, X.; Lu, R. Poisoning and Evasion Attacks against Deep Learning Algorithms in Autonomous Vehicles. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4439–4449. [CrossRef]
45. Chen, S.T.; Cornelius, C.; Martin, J.; Chau, D.H.P. ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2018 ECML PKDD 2018, Dublin, Ireland, 10–14 September 2018.
46. Zhao, Y.; Zhu, H.; Liang, R.; Shen, Q.; Zhang, S.; Chen, K. Seeing isn't Believing: Towards More Robust Adversarial Attack against Real World Object Detectors. In Proceedings of the ACM Conference on Computer and Communications Security, London, United Kingdom, 11–15 November 2019.
47. Yang, W.; Zhang, Y.; Chen, G.; Yang, C.; Shi, L. Distributed Filtering under False Data Injection Attacks. *Automatica* **2019**, *102*, 34–44. [CrossRef]
48. Sun, H.-T.; Peng, C.; Ding, F. Self-Discipline Predictive Control of Autonomous Vehicles against Denial of Service Attacks. *Asian J. Control* **2022**, *24*, 3538–3551. [CrossRef]
49. Zhao, C.; Gill, J.S.; Pisu, P.; Comert, G. Detection of False Data Injection Attack in Connected and Automated Vehicles via Cloud-Based Sandboxing. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 9078–9088. [CrossRef]
50. Hosseinzadeh, M.; Sinopoli, B. Active Attack Detection and Control in Constrained Cyber-Physical Systems Under Prevented Actuation Attack. In Proceedings of the 2021 American Control Conference (ACC), New Orleans, LA, USA, 25–28 May 2021.
51. Zhou, H.; Li, W.; Kong, Z.; Guo, J.; Zhang, Y.; Yu, B.; Zhang, L.; Liu, C. Deepbillboard: Systematic Physical-World Testing of Autonomous Driving Systems. In Proceedings of the International Conference on Software Engineering, Seoul, Republic of Korea, 27 June–19 July 2020.
52. Boloor, A.; Garimella, K.; He, X.; Gill, C.; Vorobeychik, Y.; Zhang, X. Attacking Vision-Based Perception in End-to-End Autonomous Driving Models. *J. Syst. Archit.* **2020**, *110*, 101766. [CrossRef]
53. Tampuu, A.; Matiisen, T.; Semikin, M.; Fishman, D.; Muhammad, N. A Survey of End-to-End Driving: Architectures and Training Methods. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *33*, 1364–1384. [CrossRef] [PubMed]
54. Chitta, K.; Prakash, A.; Jaeger, B.; Yu, Z.; Renz, K.; Geiger, A. TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 12878–12895. [CrossRef]

55. Pérez-Gil, Ó.; Barea, R.; López-Guillén, E.; Bergasa, L.M.; Gómez-Huélamo, C.; Gutiérrez, R.; Díaz-Díaz, A. Deep Reinforcement Learning Based Control for Autonomous Vehicles in CARLA. *Multimed. Tools Appl.* **2022**, *81*, 3553–3576. [CrossRef]
56. Ye, F.; Zhang, S.; Wang, P.; Chan, C.Y. A survey of Deep Reinforcement Learning Algorithms for Motion Planning and Control of Autonomous Vehicles. *IEEE Intell. Veh. Symp. Proc.* **2021**, *2021*, 1073–1080. [CrossRef]
57. Pomerleau, D.A. Alvinn: An Autonomous Land Vehicle in a Neural Network. *Adv. Neural Inf. Process. Syst.* **1989**, *1*, 305–313.
58. Eraqi, H.M.; Moustafa, M.N.; Honer, J. End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies. *arXiv* **2017**, arXiv:1710.03804.
59. George, L.; Buhet, T.; Wirbel, E.; Le-Gall, G.; Perrotton, X. Imitation Learning for End to End Vehicle Longitudinal Control with Forward Camera. *arXiv* **2018**, arXiv:1812.05841.
60. Chen, Z.; Huang, X. End-To-end Learning for Lane Keeping of Self-Driving Cars. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Redondo Beach, CA, USA, 11–14 June 2017.
61. Bai, T.; Luo, J.; Zhao, J.; Wen, B.; Wang, Q. Recent Advances in Adversarial Training for Adversarial Robustness. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Montreal-themed Virtual Reality, 19–26 August 2021.
62. Wong, E.; Rice, L.; Kolter, J.Z. Fast is better than free: Revisiting adversarial training. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
63. Klingner, M.; Kumar, V.R.; Yogamani, S.; Bar, A.; Fingscheidt, T. Detecting Adversarial Perturbations in Multi-Task Perception. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Kyoto, Japan, 23–27 October 2022.
64. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, 23–25 May 2016.
65. Liu, Z.; Liu, Q.; Liu, T.; Xu, N.; Lin, X.; Wang, Y.; Wen, W. Feature Distillation: DNN-Oriented jpeg Compression against Adversarial Examples. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
66. Pang, T.; Xu, K.; Du, C.; Chen, N.; Zhu, J. Improving Adversarial Robustness via Promoting Ensemble Diversity. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 10–15 June 2019.
67. Yan, Z.; Guo, Y.; Zhang, C. Deep defense: Training DnNs with Improved Adversarial Robustness. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018.
68. Lecuyer, M.; Atlidakis, V.; Geambasu, R.; Hsu, D.; Jana, S. Certified Robustness to Adversarial Examples with Differential Privacy. In Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA, USA, USA, 19–23 May 2019.
69. Zheng, Z.; Hong, P. Robust Detection of Adversarial Attacks by Modeling the Intrinsic Properties of Deep Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018.
70. Lee, K.; Lee, K.; Lee, H.; Shin, J. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018.
71. Wang, R.; Chen, Z.; Dong, H.; Xuan, Q. You Can't Fool All the Models: Detect Adversarial Samples via Pruning Models. *IEEE Access* **2021**, *9*, 163780–163790. [CrossRef]
72. Chen, B.; Carvalho, W.; Baracaldo, N.; Ludwig, H.; Edwards, B.; Lee, T.; Molloy, I.; Srivastava, B. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In Proceedings of the AAAI Workshop on Artificial Intelligence Safety, Honolulu, HI, USA, 27 January 2019.
73. Jin, G.; Shen, S.; Zhang, D.; Dai, F.; Zhang, Y. APE-GAN: Adversarial Perturbation Elimination with GAN. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, UK, 12–17 May 2019.
74. Udacity A Self-Driving Car Simulator Built with Unity. Available online: https://github.com/udacity/self-driving-car-sim (accessed on 16 May 2021).
75. Bruck, L.; Haycock, B.; Emadi, A. A Review of Driving Simulation Technology and Applications. *IEEE Open J. Veh. Technol.* **2021**, *2*, 1–16. [CrossRef]
76. Hussain, M.; Ali, N.; Hong, J.E. DeepGuard: A Framework for Safeguarding Autonomous Driving Systems from Inconsistent Behaviour. *Autom. Softw. Eng.* **2022**, *29*, 1–32. [CrossRef]
77. Hussain, M.; Ali, N.; Hong, J.-E. Vision Beyond the Field-of-View: A Collaborative Perception System to Improve Safety of Intelligent Cyber-Physical Systems. *Sensors* **2022**, *22*, 6610. [CrossRef] [PubMed]
78. Shibly, K.H.; Hossain, M.D.; Inoue, H.; Taenaka, Y.; Kadobayashi, Y. Towards Autonomous Driving Model Resistant to Adversarial Attack. *Appl. Artif. Intell.* **2023**, *37*, 2193461. [CrossRef]
79. Li, Y.; Velipasalar, S. Weighted Average Precision: Adversarial Example Detection in the Visual Perception of Autonomous Vehicles 2020. *arXiv* **2020**, arXiv:2002.03751.
80. Haq, F.U.; Shin, D.; Nejati, S.; Briand, L. Comparing Offline and Online Testing of Deep Neural Networks: An Autonomous Car Case Study. In Proceedings of the 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, 24–28 October 2020; pp. 85–95. [CrossRef]