UniAda: Universal Adaptive Multiobjective Adversarial Attack for End-to-End Autonomous Driving Systems

Jingyu Zhang, Jacky Wai Keung, *Senior Member, IEEE*, Yan Xiao, Yihan Liao, Yishu Li, *Graduate Student Member, IEEE*, and Xiaoxue Ma

***Abstract*—Adversarial attacks play a pivotal role in testing and improving the reliability of deep learning (DL) systems. Existing literature has demonstrated that subtle perturbations to the input can elicit erroneous outcomes, thereby substantially compromising the security of DL systems. This has emerged as a critical concern in the development of DL-based safety–critical systems like autonomous driving systems (ADSs). The focus of existing adversarial attack methods on end-to-end (E2E) ADSs has predominantly centered on misbehaviors of steering angle, which overlooks speed-related controls or imperceptible perturbations. To address these challenges, we introduce UniAda–a multiobjective white-box attack technique with a core function that revolves around crafting an image-agnostic adversarial perturbation capable of simultaneously influencing both steering and speed controls. UniAda capitalizes on an intricately designed multiobjective optimization function with the adaptive weighting scheme (AWS), enabling the concurrent optimization of diverse objectives. Validated with both simulated and real-world driving data, UniAda outperforms five benchmarks across two metrics, inducing steering and speed deviations from 3.54° to 29° and 11 to 22 km/h on average. This systematic approach establishes UniAda as a proven technique for adversarial attacks on modern DL-based E2E ADSs.**

***Index Terms*—Adversarial attacks, autonomous driving, deep learning (DL), multiobjective optimization, white-box attacks.**

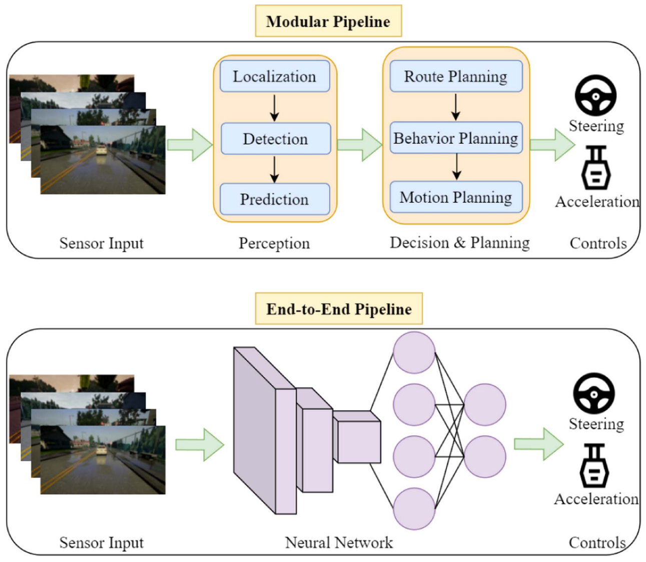

Fig. 1. Overview of modular ADS pipeline (top) and E2E DL-based ADS pipeline (bottom).

I. Introduction

AUTONOMOUS driving has brought about a transformative shift in the driving experience, showcasing the immense potential to significantly reduce accidents attributed to human errors and alleviate traffic congestion challenges [1].

Manuscript received 11 September 2023; revised 12 January 2024 and 22 March 2024; accepted 18 April 2024. This work was supported in part by the General Research Fund (GRF) of the Research Grants Council of Hong Kong; in part by the industry research funds of City University of Hong Kong under Grant 7005217, Grant 9220097, Grant 9220103, Grant 9229029, Grant 9229098, and Grant 9678149; in part by the National Natural Science Foundation of China under Grant 62025604; and in part by Shenzhen Science and Technology Program under Grant KQTD20221101093559018. Associate Editor: L. Ma. *(Corresponding author: Yan Xiao.)*

Jingyu Zhang, Jacky Wai Keung, Yihan Liao, Yishu Li, and Xiaoxue Ma are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: jzhang2297-c@my.cityu.edu.hk; Jacky.Keung@cityu.edu.hk; yihanliao4-c@my.cityu.edu.hk; yishuli5-c@my.cityu.edu.hk; xiaoxuema3-c@my.cityu.edu.hk).

Yan Xiao is with the School of Cyber Science and Technology, Sun Yat-Sen University, Shenzhen 510275, China (e-mail: xiaoy367@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TR.2024.3394894

There are two main approaches to implementing the autonomous driving system (ADS): modular [2] and End-to-End (E2E) [3]. Fig. 1 gives a simple architecture overview of these two systems. The conventional modular approach decomposes the system into interconnected modules encompassing perception, decision-making, planning, and control [2], [4], [5]. Recent advancements in deep learning (DL) have spurred interest in E2E methods, which combine multiple modules into a single deep neural network [usually convolutional neural network (CNN)-based] model. E2E ADSs predominantly input sensor data (e.g., RGB images) and directly output numerical control actions such as steering and acceleration. While E2E ADSs have exhibited remarkable success [6], [7], [8], [9], they remain susceptible to malicious input data, including misleading corner cases or adversarial examples [10], [11], [12], [13].

Existing studies have delved into generating transformed or adversarial images as inputs to reveal the misleading prediction of the ADS under test [10], [11], [12], [13]. Usually, transformed images result from noticeable transformations applied to original inputs. Adversarial examples [14], [15], [16], [17], [18] involve subtle imperceptible perturbations to original data, avoiding human detection. These studies explore diverse transformations or perturbations to the input images and analyze their impact on steering actions. These range from modifying weather conditions [11], [19], [20] to introducing small black occlusions [13]. However, these efforts fall short in comprehensively testing ADSs, marked by three limitations.

1) *Limited Vehicle Controls Testing:* Existing studies [10], [11], [12], [21] mainly concentrate on attacking steering control, while neglecting speed-related controls. This falls short of testing the security of real-world driving. Notably, no literature exists on targeting image inputs solely to trigger errors in both steering and acceleration controls for E2E ADSs based on DL.
2) *Constrained Testing Scenarios:* Current testing scenarios lack diversity. For instance, SelfOracle proposed by Stocco et al. [22] exclusively focuses on highway scenarios, excluding intersections, traffic lights, pedestrians, and other vehicles. Similarly, attack techniques like DeepBillboard [10] and PhysGAN [12] concentrate on driving scenarios containing billboards. The state-of-the-art technique, DeepManeuver [21], exclusively perform attacks within a simulator, lacking the testing of real-world scenarios.
3) *Image-specific or Unnatural Perturbations:* Many existing techniques generate perturbations or transformations specific to individual images [11], [13], [19], [20]. Alternatively, they produce perturbations easily detectable by humans, such as unnatural billboard replacements [10], [12], [21]. These limitations prompted our exploration of an adversarial perturbation–a subtle, human-imperceptible modification capable of consistently inducing errors in both speed and steering controls across a sequence of images (e.g., a driving record).

In addressing these shortcomings, we design UniAda that leverages adaptive multiobjective learning to generate universal adversarial perturbations for ADSs. Notably, we term an adversarial perturbation as *universal* when it consistently triggers errors larger than a predefined threshold across most images sampled from the data distribution [23], [24], thereby transcending image-specific constraints. There could still exist a limited number of images within the distribution where the generated universal perturbation fails to trigger errors surpassing the predefined threshold. In this article, we focus on the case where the distribution represents the set of images describing a driving scenario (e.g., car stops for a jaywalking pedestrian). To demonstrate the universality, we report the percentage of images that our generated perturbation attacks (PAs) successfully by the success rate metric with four different tested thresholds for both objectives. The results have shown that UniAda can successfully attack 96.3% images with steering error larger than 3.5°, which is a substantial improvement compared to baselines.

More specifically, UniAda optimizes a multiobjective function during perturbation generation. To achieve universality, UniAda establishes a joint optimization function through the summation of multiobjective functions associated with each image within the input driving record.

Through iterative gradient descent, the joint function is minimized, facilitating the search for the perturbation. The resultant perturbation consistently triggers errors across multiple model predictions for most input images.

Departing from rudimentary uniform weighting or exhaustive grid search for objective weights, we propose the adaptive weighting scheme (AWS), inspired by Chen et al. [25]. AWS, a gradient-based weight adjustment method, strives to balance varying objective weights to ensure all objectives are trained at a similar rate.

This empowers UniAda to iteratively determine the optimal weight combination, enhancing the acquisition of shared feature representations for diverse objectives. To test the effectiveness of AWS, we proposes a variant of UniAda, dubbed UniEqual, which shares the identical algorithm concepts but omits the use of AWS. We also conduct statistically analysis (i.e., t-test) to intuitively show the difference between UniAda and UniEqual, which implies that AWS brings a statistically significant improvement in most cases.

We evaluate UniAda on 14 urban driving videos selected from four datasets (i.e,, Carla100 [8], Kitti [26], Udacity [9], Dave [27]) for two objectives (Steering and Acceleration) with three victim ADSs [i.e, CILRS, CILR, MotionTransformer (MT)]. Our results demonstrate UniAda's superiority over five benchmarks (DeepManeuver [21], DeepBillboard [10], PA [28], FGSM [15], UniEqual) in terms of key evaluation metrics-mean error (ME) and success rate (SR). Notably, UniAda achieves the highest ME in steering (29.2° for CILRS, 25.6° for CILR, and 3.54° for MT), which is 6.6°, 8.6°, and 2.93° higher than the single-objective state-of-the-art method DeepManeuver. Furthermore, compared to multiobjective techniques (i.e., PA, FGSM and UniEqual), UniAda emerges as the most effective solution for both objectives on average for all three models in almost all cases.

We summarize the key contributions of this article as follows.

1) We propose UniAda, a novel attack technique for DL-based E2E ADSs that can generate multiobjective universal adversarial perturbations for the input driving video. UniAda is effective in triggering errors on three state-of-the-art DL-based E2E ADSs for both steering and acceleration controls tested with both simulated and real-world data. To the best of our knowledge, we are the first to conduct imperceptible image-agnostic offline attacks to induce multiobjective misbehaviors for E2E ADSs.
2) We propose a new strategy, AWS, employed in UniAda for balancing different objectives in a real-time manner, which maintains consistent training rates for each objective. The effectiveness of AWS is validated by comparing UniAda with the equal-weighted counterpart, UniEqual.
3) We validate the effectiveness of the multiobjective attack, by comparing it with single-objective counterparts. In addition, we assess UniAda's effectiveness in urban traffic under both simulated and real-world driving environment.

II. Related Work

A. DL in Autonomous Driving

The integration of DL into E2E autonomous systems [29], [30], [31] traces back to the late 1980 s, Pomerleau et al. [32] built the autonomous land vehicle in a neural network (ALVINN) system, which uses a shallow three-layer network. Later in 2016, Bojarski et al. [6] trained a CNN to directly map raw image pixels from a single front-facing camera to steering commands. Similarly, in the context of the Udacity self-driving car challenges [9], researchers embarked on constructing cutting-edge agents that interpret sensor inputs to govern vehicle operations. More recently, Codevilla et al. [8] introduced an innovative conditional imitation learning framework, which takes RGB images, speed measurements and navigation instructions as inputs, generating steering and acceleration-related car controls as outputs.

B. Attack DL Systems

In the realm of contemporary software development, the widespread integration of DL has yielded satisfactory outcomes. However, extensive research [12], [15], [17], [33], [34], [35], [36], [37], [38], [39] has demonstrated that many DL models remain susceptible to intentional adversarial attacks designed to provoke misbehavior in these models. A prevalent form of such attacks involves the use of adversarial examples, characterized by imperceptible or quasi-imperceptible perturbations from the original inputs. By feeding the adversarial examples to the DL model, the model can produce different predictions from the original ones. This explores and reveals the vulnerabilities of the DL systems.

Out of all the vulnerabilities, the *offline white-box attack* poses the most significant threat to DL systems, due to the full access to the targeted model and potential iterative interactions with it [40].

Both offline attacks and white-box attacks are categorized within the realm of adversarial attacks. Offline attacks [15], [28], [41], [42] apply the adversarial perturbation on a preacquired fixed dataset, where the attack performance is commonly evaluated by the prediction error between original and adversarial prediction. Specifically, offline attacks exploit the vulnerabilities without requiring a real-time interaction with the target system, allowing attackers to carefully analyze and manipulate models to design the attack technique.

In the field of white-box attacks [43], [44], information regarding internal model structures and parameters are available for the attack techniques to trigger erroneous behavior, primarily beneficial for software verification. This in-depth understanding enables the attacker to formulate highly targeted and effective adversarial perturbations to manipulate the model's behavior. In the context of attacking DNNs, white-box attacks normally modify original inputs by using gradients computed with respect to relevant metrics, thereby inducing erroneous prediction (e.g., misclassification). For example, FGSM [15] utilizes gradient-based white-box attacks in image classification tasks such that the perturbed image will be wrongly classified, e.g., a perturbed dog image will be misclassified as cat by the DNN model with high accuracy.

In this article, we focus on an offline white-box gradient-based attack strategy, which generates adversarial perturbations aimed at inducing misbehavior in DNN-based E2E ADS.

C. Offline Attacking ADSs

In the domain of offline attacks on DNN-based ADSs, a plethora of methodologies has been proposed. Many of these approaches, such as [10], [11], [12], [13], and [19], are designed to generate adversarial images as sensor inputs, with the objective of triggering errors in the targeted ADSs. These methods seek transformations from original images that result in transformed or adversarial images, capable of inducing model misbehavior. For instance, DeepXplore [13] employs neuron coverage and differential behavior-based metrics within a white-box testing framework to create transformed images that mislead the steering angle predictions of ADSs. These transformed images might exhibit different lighting conditions or incorporate small black occlusions when compared to the originals.

The aforementioned studies primarily focus on image-specific transformations, in which they apply distinct transformations to different images, rendering a single transformation ineffective across multiple images. In recent work, some studies [10], [12], [21], [45] have pushed beyond this constraint by seeking transformations capable of consistently inducing model misbehavior, affecting sequences of images (i.e., driving records). Notably, Von et al. (DeepManeuver) [21], Zhou et al. (DeepBillboard) [10], and Kong et al. (PhysGAN) [12] utilized variants of prediction differences as target metrics to generate adversarial perturbations. These methods identify a universal perturbation pattern that can persistently provoke misbehavior on steering predictions across a sequence of driving images. However, their focus is confined to specific testing scenarios involving billboard appearances and targeting steering angle misbehavior exclusively. Moreover, DeepManeuver exclusively performs attacks within a simulator.

The majority of these techniques concentrate on single-objective attacks on E2E ADSs, primarily aiming at misleading steering predictions. Beyond steering angle, speed-related controls bear equal significance in driving scenarios. This article introduces a multiobjective attack strategy for E2E ADSs, generating perturbations that remain imperceptible to human eyes, yet consistently elicit model misbehavior across multiple controls. We also propose an AWS to balance the influence of each objective, ensuring comparable training rates, and higher efficiency in finding complementary perturbation patterns.

III. Preliminaries

Adversarial Attacks: In this section, we provide a concise and formulated overview of adversarial attacks, covering both image classification and regression problems. In addition, we elucidate the concept of universal perturbation.

In adversarial attacks, the generated perturbation τ can trigger errors in the model prediction. By adding a subtle τ to the input sample x (e.g., an RGB image), a model m trained with optimal parameter μ can make a different prediction (mostly wrong) from the original. Specifically, for image classification problems, we wish to find a vector τ that can induce the CNN model to predict different class labels for x and $x + \tau$, satisfying

$$m(x+\tau;\mu) \neq m(x;\mu) \qquad \text{s.t. } \|\tau\|_{\mathrm{p}} \leq \epsilon \tag{1}$$

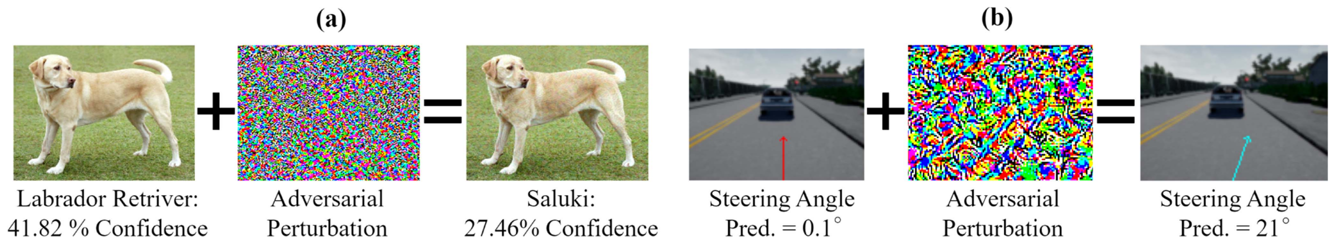
(a)
Labrador Retriver: 41.82 % Confidence
Adversarial Perturbation
Saluki: 27.46% Confidence
(b)
Steering Angle Pred. = 0.1°
Adversarial Perturbation
Steering Angle Pred. = 21°

Fig. 2. Examples of adversarial attacks on image classification and regression tasks. Pixel values of perturbation τ are scaled for visibility. (a) Adversarial attacks on classification problems. (b) Adversarial attacks on regression problems.

where $\|\cdot\|_p$ indicates the L_p norm, and L_0, L_2, and $L_{\inf}$ are commonly used metrics [15], [24]. The constraint is to ensure $x+\tau$ and x are visually the same to human eyes. For regression problems, the goal of the adversarial attack is to find τ that maximizes the discrepancy between the two numerical predictions, satisfying

$$\underset{\tau}{\text{argmax}}\ |m(x+\tau;\mu) - m(x;\mu)| \qquad \text{s.t. } \|\tau\|_{\text{p}} \leq \epsilon. \tag{2}$$

Normally, in regression problems, the attack is considered successful if the prediction difference is larger by a predefined threshold [46].

Fig. 2 gives examples of adversarial attacks on the image classification model [see Fig. 2(a) with $\epsilon = 15$] and autonomous driving regression model [see Fig. 2(b) with $\epsilon = 2$]. Specifically, for the classification task, the CNN model is very vulnerable to a subtle perturbation, which changes its prediction from *Labrador Retriver* with 41.82% confidence to *Saluki* with 27.46% confidence. For the regression task, we show an example of a CNN-based E2E autonomous driving model that predicts the steering angle value from the image. The added imperceptible perturbation changes its prediction from 0.1° to 21°. The above example shows that CNN models are vulnerable to such attacks.

For a *Universal Perturbation* [24] (i.e., image-agnostic), the goal is to seek a numerical vector τ that can successfully fool almost all datapoints sampled from $\mathbf{X}$, where $\mathbf{X}$ denotes a distribution of data. In regression problems, a universal perturbation τ satisfies

$$\begin{aligned} &|m(x+\tau;\mu) - m(x;\mu)| > \delta \\ &\text{s.t. } \|\tau\|_p \leq \epsilon \qquad for\ most\ x \in \mathbf{X}. \end{aligned} \tag{3}$$

In this article, $\mathbf{X}$ is a distribution of images, represented by a driving record. We attack three CNN-based autonomous driving models (introduced in Section VI-B2) that perform a multioutput regression task, which takes an RGB image as input and outputs continuous numerical steering angle and acceleration prediction.

IV. Threat Model

Our work focuses on attacking DNN-based autonomous driving models in the image regression domain. We consider four types of threat model in this work: adversarial falsification, adversary's knowledge, adversarial specificity, and attack frequency [40].

Adversarial Falsification: For intentional attack, the attacker can design an adversarial example $(x+\tau)$ to fool the model m into making a wrong decision. However, this manipulated example is not visually different from x to a human observer. For unintentional attack, the attacker can unintentionally send an input $\hat{x}$ that is sampled from a distribution different from the training distribution. Specifically, $\hat{x}$ has a label y, and $m(\hat{x}) \neq y$. In this case, the model fails on $\hat{x}$ due to the distribution shift.

Our goal is to perform *intentional* attack on regression problem, which adds a subtle perturbation τ on sample image x to cause misbehavior on autonomous driving model m. The adversarial image is similar to the original image and conforms to the training distribution.

Adversary's Knowledge: Based on the available knowledge to the model, attacks can be categorized into white-box, grey-box, and black-box. They correspond to full access, partial access, and no access to the model internal structure and parameter information. In this article, we perform *white-box* gradient-based attacks, where we know the model architecture, hyperparameters, model weights. We generate adversarial examples by calculating model gradients with our designed objective function.

Adversarial Specificity: This includes targeted attacks and nontargeted attacks. For targeted attacks, the attacker misguides DNN to a specified prediction output (e.g., specified predicted class for classification problem, bigger/smaller predicted values for regression problems). In nontargeted attacks, the adversarial output can be arbitrary except the original one.

We construct a *targeted* adversarial sample that follows specified attack direction. For example, we ask UniAda to construct an adversarial sample that can induce the model to turn right (i.e., to predict bigger steering value) instead of turn left.

Attack Frequency: This contains one-step attacks and iterative attacks, which differs from the number of interactions with the victim model. We use *iterative* attacks that take multiple times to update the adversarial examples to reach a better performance.

V. Methodology

A. UniAda Overview

This section outlines UniAda, which employs adaptive objective weights to create a universal adversarial perturbation. This perturbation consistently misleads the targeted model across multiple images, impacting several car controls simultaneously. The overview of UniAda is shown in Fig. 3. Given a driving video encompassing dozens of images and initial objective weights,

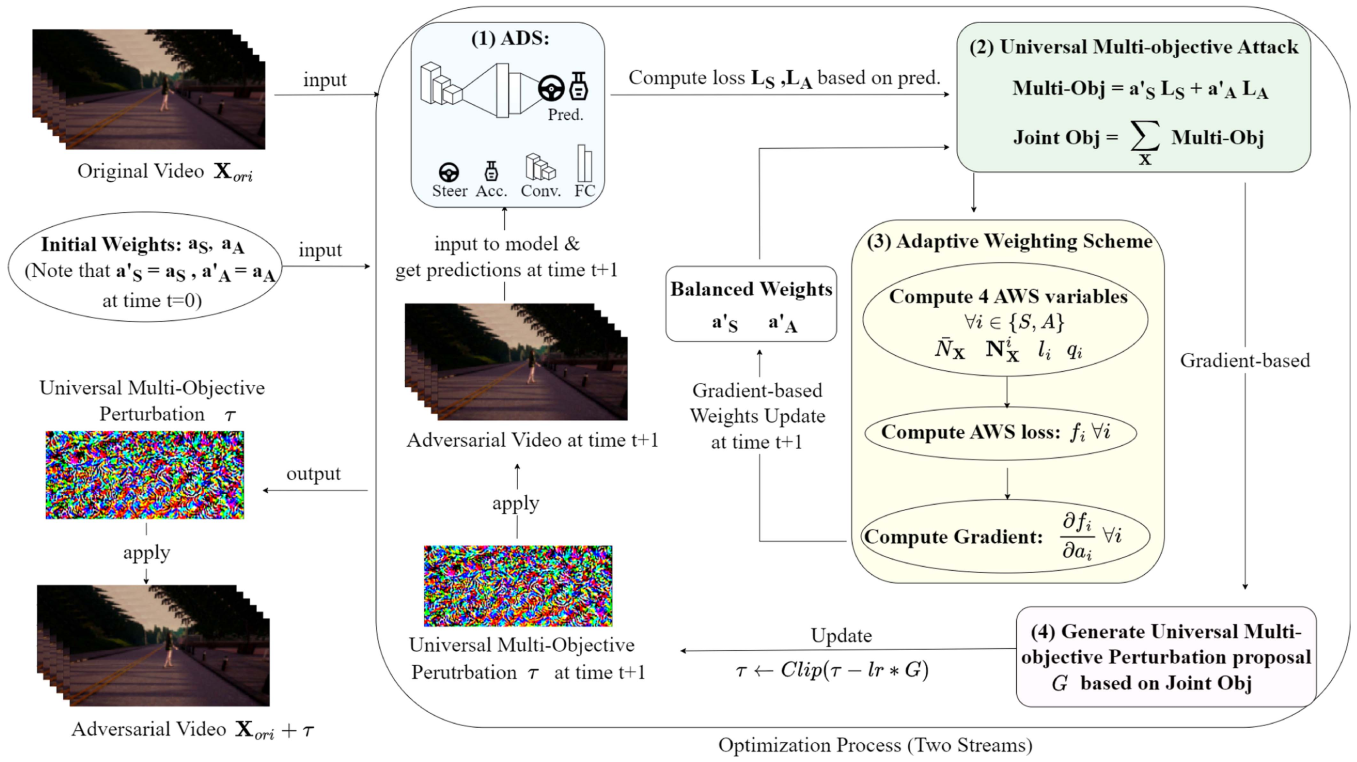
Original Video $\mathbf{X}_{ori}$
input
(1) ADS:
Pred.
Steer Acc. Conv. FC
Compute loss $\mathbf{L_S}$, $\mathbf{L_A}$ based on pred.
(2) Universal Multi-objective Attack
Multi-Obj = $\mathbf{a'_S L_S + a'_A L_A}$
Joint Obj = $\sum_{\mathbf{X}}$ Multi-Obj
Initial Weights: $\mathbf{a_S}$, $\mathbf{a_A}$
(Note that $\mathbf{a'_S = a_S}$, $\mathbf{a'_A = a_A}$ at time t=0)
input
input to model & get predictions at time t+1
Balanced Weights $\mathbf{a'_S}$ $\mathbf{a'_A}$
(3) Adaptive Weighting Scheme
Compute 4 AWS variables $\forall i \in \{S, A\}$
$\bar{N}_{\mathbf{X}}$ $\mathbf{N}^i_{\mathbf{X}}$ l_i q_i
Compute AWS loss: f_i $\forall i$
Compute Gradient: $\frac{\partial f_i}{\partial a_i}$ $\forall i$
Gradient-based
Universal Multi-Objective Perturbation τ
output
Adversarial Video at time t+1
Gradient-based Weights Update at time t+1
apply
apply
Adversarial Video $\mathbf{X}_{ori} + \tau$
Universal Multi-Objective Perutrbation τ at time t+1
Update
$\tau \leftarrow Clip(\tau - lr * G)$
(4) Generate Universal Multi-objective Perturbation proposal G based on Joint Obj
Optimization Process (Two Streams)

Fig. 3. UniAda Overview: Our attack technique contains two main optimization streams. The first one is the universal multiobjective attack (in green) to generate a perturbation that is influential to multiple images and multiple car controls (i.e., steering S and acceleration A). The second one is the AWS (in yellow), which iteratively balances the objective weights to ensure similar training rates. Best viewed in color.

UniAda commences the perturbation discovery through parallel optimization streams: perturbation optimization and objective weight optimization. These streams evolve in tandem during each iteration step t.

In the perturbation optimization stream, each image inside the input video is fed into the autonomous driving model. This yields predictions for each car control, specifically steering angle and acceleration, as depicted in step *(1) ADS*. Utilizing these predictions, UniAda computes individual losses for each car control (L_S, L_A). Drawing from these losses and the prevailing objective weights a'_S, a'_A, UniAda in step *(2)* computes the joint optimization function (detailed explanations can be found in Section V-B). Subsequently, employing a gradient-based methodology, UniAda derives the joint function's derivative with respect to the image, and the resultant gradient G serves as the current perturbation proposal. G is then leveraged to update the universal multiobjective perturbation for the subsequent time step $t+1$.

Simultaneously, the weight optimization procedure starts, enabling the dynamic adjustment of weights for each objective. At a given step t, UniAda employs the ongoing objective weights a'_S, a'_A, coupled with the losses (L_S, L_A), to calculate four variables within step *AWS*. These variables contribute to the AWS loss. Subsequently, UniAda computes gradients of the AWS loss f_i with respect to each objective weight a_i. These gradients facilitate a gradient-based update mechanism, resulting in refreshed balanced weights at the following time step. Details for AWS are elaborated in Section V-C.

To generate universal multiobjective perturbations, both optimization processes iteratively proceed until the maximum number of epochs is reached.

B. Universal MultiObjective Attack

We design the loss function [12] for each objective i to maximize the difference between original and adversarial model predictions

$$L_i(\mathbf{X}^n_{ori}, \tau) = \frac{1}{\beta}\exp\left\{-\frac{1}{\beta}(m_i(\mathbf{X}^n_{ori}+\tau)-m_i(\mathbf{X}^n_{ori})) \times d_i\right\}$$
$$\forall n \in \{1, \ldots, N\}. \quad (4)$$

The hyperparameter $d_i = \pm 1$ is the attack direction for objective (car control) i, which is set to be $-1/1$ if we would like to attack in the negative/positive direction (adversarial prediction is learned to be smaller/larger than the original).

Notations encompass τ for perturbation at time t, $\mathbf{X}^n_{\text{ori}}$ signifying the nth image frame inside the input video $\mathbf{X}_{\text{ori}}$, and $m_i(\mathbf{X}^n_{\text{ori}} + \tau)$ representing the prediction of autonomous driving model m for objective i from the perturbed adversarial image $\mathbf{X}^n_{\text{ori}} + \tau$. β signifies the sharpness parameter. N denotes the total number of images of the input driving record. Similar to DeepBillboard, we use original prediction as the attack reference.

Notably, prior work solely concentrated on perturbations inducing deviations in the model's steering angle predictions, disregarding speed-related car control objectives. To generate a universal perturbation capable of concurrently influencing multiple car controls, we devise a multiobjective loss as follows:

$$O(\mathbf{X}^n_{\text{ori}}, \tau) = \sum_{i=1}^{C} a_i L_i(\mathbf{X}^n_{\text{ori}}, \tau) \quad \forall n \in \{1, \ldots, N\} \tag{5}$$

where C denotes the number of objectives, and parameter a_i reflects the weight assigned to objective i. In our case, there are two objectives, steering and acceleration, denoted by $i = S$ and $i = A$, respectively. Diverging from a fixed a_i, UniAda introduces the AWS for real-time updates to a_i, effectively balancing each objective's contribution.

To generate a universally applicable perturbation that consistently impacts all input images, UniAda find τ by optimizing the subsequent joint optimization function

$$\underset{\tau}{\text{argmin}} \sum_{n=1}^{N} O(\mathbf{X}^n_{\text{ori}}, \tau). \tag{6}$$

The generation of perturbation integrates the joint objective for all image frames within the video. Utilizing a gradient-based technique, UniAda minimizes this joint function, iteratively refining τ.

C. Adaptive Weighting Scheme

During the multiobjective optimization process $O = \sum_i^C a_i L_i$ [as depicted in (5)], it is critical to select an appropriate weight for each objective. In our context, they significantly affect the contribution of each individual objective to the perturbation proposal. In the simplest case, one may opt for equal weighted objectives, where each objective is assigned a weight of $a_i = \frac{1}{C}$. In this case, the optimization process may not capture the importance of each objective accurately. Another option involves using computationally expensive grid search methods [47], which systematically explore a predefined set of values for each objective weight and select the combination that maximizes performance. However, it becomes computationally impractical for problems with a large parameter space. In our case, we employ an adaptive method [25] dubbed AWS, where a_i can vary at each time step t during the perturbation optimization process. AWS is a complementary component integrated into the multiobjective optimization process, allowing us to determine the optimal value for each a_i at each time step t. This ensures a balance in the contribution of each objective for optimal perturbation searching. Specifically, AWS updates each objective weight based on its gradient to the AWS loss [see (7)] at each step t. AWS operates by assigning larger weights to objectives contributing less (indicating undertraining) and smaller weights to those that are overly trained. Through AWS weight updates, both objectives collectively contribute to the perturbation, ensuring effective triggering of errors in both steering and acceleration controls, rather than confining influence to just one.

The AWS loss is formulated to capture differences between the weighted gradient norm for each objective i (reflecting objective i's perturbation pattern's impact on the universal perturbation) and the target norm (representing the desired influence of objective i's perturbation on the universal perturbation). This target influence considers weighted gradient norms of other objectives ($\overline{N}_{\hat{X}}$) and the extent of misbehavior in objective i compared to other objectives $(q_i)^\gamma$. The AWS loss f_i for each i is expressed as follows:

Algorithm 1: Adaptive Weighting Scheme.

```
Require: $\hat{X}$ - mini-batch images at search time $t$
Require: $a_i$ - the weight of objective $i$ at time $t$
Require: $L_i^0$ - initial loss for objective $i$
Require: $lr_{grad}$ - hyperparameter, learning rate for AWS
   update
Require: $\gamma$ - hyperparameter, strength of rebalance
1: for $x$ in $\hat{X}$ do
2:     Compute $L_i(x)$, $\frac{\partial L_i(x)}{\partial x}$ $\forall i$
3: end for
4: Compute AWS variables $N^i_{\hat{X}}, \overline{N}_{\hat{X}}, l_i, q_i, f_i^{grad}$ $\forall i$ at
   time $t$
5: Conditionally Update $a_i \leftarrow a_i - lr_{grad} \times f_i^{grad}$
6: Normalize $\sum_i a_i = 1$
```

$$f_i = \left| N^i_{\hat{X}} - \overline{N}_{\hat{X}} \times (q_i)^\gamma \right| \tag{7}$$

where f_i consists of four variables, $N^i_{\hat{X}}$, $\overline{N}_{\hat{X}}$, l_i, q_i, with l_i utilized in the computation of q_i. These four variables represent the objective-specific norm, objective-average norm, the loss ratio that captures the inverse training rate for i, and the relative inverse training rate for i, respectively. The hyperparameter γ controls the strength of rebalance; larger γ exerts greater restoring force to pull objectives back to similar training rates. Detailed formulae for each variable are presented as follows.

1. $N^i_{\hat{X}} = E_{\hat{X}}[\|a_i \frac{\partial L_i(x)}{\partial x}\|_2]$: the objective-specific norm.[1] $\hat{X} \subseteq \mathbf{X}_{\text{ori}} + \tau$ denotes the mini-batch image set at the time t, and $x \in \hat{X}$. $N^i_{\hat{X}}$ denotes the average of weighted gradient norms over all images inside the mini-batch at time t. We treat each image equally when computing the average.

2. $\overline{N}_{\hat{X}} = E_i[N^i_{\hat{X}}]$: an objective-average norm that averages the objective-specific norms across all objectives.

3. $l_i = \frac{E_{\hat{X}}[L_i(x)]}{L_i^0}$: the loss ratio that captures the inverse training rate for the objective i. A smaller value indicates a higher training rate. Numerator is the batch average loss for objective i at time t. The denominator denotes the initial loss for the objective i, which is the mean loss of the original prediction of i at time zero over all images in the given video. The ratio is used to deal with different loss scales. With the inclusion of this variable, the objective weights are adjusted according to the misbehavior performance of i for the current mini-batch images.

4. $q_i = \frac{l_i}{E_i[l_i]}$: the relative inverse training rate for the objective i, where $E_i[l_i]$ is the mean loss ratio over different objectives. A smaller q_i gives UniAda a hint that the objective i is trained too much (i.e., loss L_i decreases too fast) compared to others, in which AWS will decrease its weight.

Algorithm 1 summarizes the AWS process. For each image, AWS computes objective loss $L_i(x)$ and its gradient (Lines

[1]$\|\cdot\|_2$ denotes Euclidean norm, it is computed after flattening the 3-D input matrix to a 1-D vector.

TABLE I
UNIADA HYPERPARAMETERS: DESCRIBE THE MEANING OF EACH HYPERPARAMETERS AND THEIR VALUES USED IN THE EXPERIMENTS

Notation	Description	Value
lr	Perturbation searching learning rate	0.2
lr_{grad}	AWS learning rate	0.005
bs	Batch size	5
β	Sharpness parameter in multi-objective loss	2
Epochs	Number of Epochs	250
γ	AWS parameter controlling rebalance strength	10
θ	Perturbation gradient rescale threshold	0.3
ϵ	Maximum perturbation to the image	{2, 5}

1–3), retaining them for AWS variable computation (Line 4). Line 5 performs objective weights update if the new weight is positive. The positive weight constraint is grounded in the necessity for $L_i(x)$ and $O(x)$ to be positively correlated, ensuring the minimization of $L_i \ \ \forall i$ during $O(x)$ minimization. Finally, the updated weights are normalized to ensure their summation equates to 1 (Line 6).

D. Generating Adversarial Perturbations

The above two sections constitute two main optimization streams: the perturbation optimization by the joint objective [see (6)] in the universal multiobjective testing, and the objective weight optimization by AWS. Both streams operate in a gradient-based manner and are iteratively updated at each step t. The algorithm details are summarized in Algorithm 2, structured across four main stages: input, initialization, perturbation searching (comprising two optimization streams), and the output stage. We elaborate on the details below.

Input: The algorithm requires four key inputs outlined in the *Require* lines. These inputs encompass the input video $\mathbf{X}_{\text{ori}}$, the autonomous driving model m responsible for predicting values across multiple car controls, the attack direction d_i corresponding to each objective i, and hyperparameters $hyper$. Table I explains the role and selected values of each hyperparameter.

Initialization: Lines 1–5 initialize all variables needed. Specifically, Line 1 creates a variable $\mathbf{X}$ to store all original image frames within the video. Line 2 initializes the universal multiobjective perturbation τ as a tensor of zeros, whose dimension is the same as the input image. Line 3 initializes objective weights a_i uniformly. Line 4 computes initial loss L_i^0 for objective i, which is used to compute the AWS variable l_i. Last, Line 5 sets the training time t as zero.

Perturbation Searching: For each epoch, UniAda shuffles all video images (Line 7) and implements learning rate decay by a predefined schedule (reduced by a factor of 0.8 every 50 epochs, as standard in the DL community [48]) to avoid local optima (Line 8). During each step t of searching, UniAda iterates over mini-batches with batch size bs (Lines 9–10). It subsequently computes single objective loss (Line 11), multiobjective loss (Line 12), and corresponding gradients with respect to each image (Line 13). Line 14 involves computation and storage of gradient norms used to determine AWS variable $N_{\hat{X}}^i$. After completing traversal across all mini-batch images in $\hat{X}$, UniAda generates gradient proposals $\frac{\partial O(x)}{\partial x}$ for each image x in the mini-batch, alongside weighted gradient norms $\|a_i \frac{\partial L_i(x)}{\partial x}\|_2$ for each image and objective i. Line 16 calculates average gradient proposals across $\hat{X}$. In instances, where gradients are too minimal to impact perturbation τ, inducing sluggish search, UniAda rescales average gradients G by $G = G \times \frac{\theta}{\|G\|_2}$ if $\|G\|_2 < \theta$ and $\|G\|_2 \neq 0$ (Line 17). This amplifies gradient magnitude, elevating its influence during addition to τ. Subsequently, Line 18 updates total perturbation τ via gradient descent, enforcing a maximum perturbation size of ϵ to maintain imperceptibility. Afterward, all image frames within $\mathbf{X}$ undergo updates, subject to processing for perceptual fidelity (Line 19). This entails confining pixel values within the range [0, 255]. Lines 20–22 encapsulate the AWS procedure, as described in Algorithm 1. These steps iteratively unfold until the maximum epoch limit is attained.

Output: Upon reaching the maximum epoch count, UniAda yields a universal multiobjective perturbation τ for the input video.

VI. EXPERIMENTS

We evaluate UniAda on three widely used multioutput autonomous driving models with 14 driving videos. The goal of our evaluation is to answer the following research questions.

A. Research Questions

RQ1. UniAda Effectiveness: How effective is UniAda in generating adversarial perturbations?

To evaluate the effectiveness of UniAda in generating perturbations that trigger errors in multiple controls, we compare its performance on 14 driving videos (7 simulated and 7 real-world videos) with four existing methods (DeepManeuver, PA, DeepBillboard, FGSM) on two evaluation metrics [Mean Error and Success Rate].

RQ2. Adaptive Weighting Scheme Effectiveness: How can the Adaptive Weighting Scheme assist the attack performance?

To assess the effectiveness of AWS, we introduce a variant of UniAda dubbed UniEqual, wherein objectives are assigned equal weights during the perturbation search process. By comparing UniEqual with UniAda, we can observe the exact impact of AWS. Furthermore, we graphically depict the trajectory of each objective weight to provide insights into the operation of AWS. We also conduct the statistical analysis (t-test) to show the significance of the improvement.

RQ3. Multiobjective Attack Effectiveness: Does the multiobjective adversarial attack improve the attack effectiveness for all objectives simultaneously compared to the single-objective attack?

Attacking multiple objectives together may yield a better result for each objective than single-objective attacks, since attacking diverse objectives can offer shared insights sometimes, yielding a more comprehensive and effective perturbation pattern. To answer this question, we compare the effectiveness of

Algorithm 2: UniAda.

```
Require: $\mathbf{X}_{ori}$ - input video
Require: $m$ - the targeted autonomous driving model
Require: $d_i \in \{+1, -1\}$ - Direction of attack: accelerate/decelerate, left/right
Require: $hyper - \{lr, lr_{grad}, bs, \beta, Epochs, \gamma, \theta, \epsilon\}$, dictionary of input hyperparameters
 1: $\mathbf{X} = copy(\mathbf{X}_{ori})$
 2: $\tau = zero(\mathbf{X}[0].shape)$
 3: $a_i = \frac{1}{C} \quad \forall i \in \{1, \ldots, C\}$
 4: Compute $L_i^0 \quad \forall i$
 5: $t = 0$
 6: for epoch in Epochs do
 7:     random.shuffle($\mathbf{X}$)
 8:     Adjust $lr$, $lr_{grad}$ for each 50 epochs
 9:     for $\hat{X}$ in $\mathbf{X}$ do
10:         for $x$ in $\hat{X}$ do
11:             $L_i(x) = \frac{1}{\beta} \exp -\frac{1}{\beta}(m_i(x) - m_i(x_{ori})) \times d_i \quad \forall i$
12:             $O(x) = \sum_{i=1}^{C} a_i L_i(x)$
13:             Compute $\frac{\partial O(x)}{\partial x}$
14:             Compute $\left\| a_i \frac{\partial L_i(x)}{\partial x} \right\|_2 \quad \forall i$
15:         end for
16:         $G = \frac{1}{len(\hat{X})} \sum_{x \in \hat{X}} \frac{\partial O(x)}{\partial x}$
17:         Conditionally Rescale $G$
18:         Update $\tau \leftarrow Clip_\epsilon(\tau - lr * G)$
19:         Update $\mathbf{X} \leftarrow Process(\mathbf{X}_{ori} + \tau)$
20:         Compute $N_{\hat{X}}^i, \overline{N}_{\hat{X}}, l_i, q_i, f_i^{grad} \quad \forall i$
21:         Conditionally Update $a_i \leftarrow a_i - lr_{grad} \times f_i^{grad}$
22:         Renormalize $\sum_i a_i = 1$
23:         $t = t + 1$
24:     end for
25: end for
26: return $\tau$
```

the adversarial perturbation generated by the multiobjective loss [see (5)] with those by single objective loss [see (4)].

B. Experimental Setup

This section provides a comprehensive overview of our experimental setup, encompassing datasets, victim autonomous driving models, baseline methods, evaluation metrics, and hyperparameters.

1) Dataset: Our experimentation is grounded in the use of 14 driving videos in total, with 7 simulated videos and 7 real-world videos. For the simulated videos, we extracted all 7 videos from the Carla100 dataset [8], which contains realistic simulated driving data (e.g., RGB images, steering and speed-related control data) captured by the central camera with 100 h of driving from an urban town of the Carla simulator (version 0.8.4). For the real-world videos, we extracted: 1) Two videos from Dave testing dataset [27], which contains 45 568 real-world driving images to test the NVIDIA Dave model; 2) Three videos from Udacity self-driving car challenge dataset [9] that contains 101 396 real-world driving images captured by a dashboard-mounted camera of a human-driven car; and 3) Two videos from Kitti dataset [26], which contains 14 999 real-world driving images from six different scenes captured from a VW Passat station wagon equipped with four cameras.

TABLE II
DESCRIPTIONS FOR THE SELECTED VIDEOS

Videos (No. of Imgs)	Description
Carla Pedestrian (18)	Stop for a pedestrian, clear sunset
Carla White Car (36)	Stop for the front white car, wet noon
Carla Black Car (22)	Stop for the front black car, clear noon
Carla Gray Car (42)	Stop for the front gray car, clear sunset
Carla Blue Car (43)	Driving Straight, following a blue car, clear sunset
Carla Red Light (66)	Stop for the front red light, wet noon
Carla Light-blue Car (21)	Stop for the front light-blue car, clear noon
Dave Curve1 (34)	Turning left at crossroad, clear noon
Dave Straight1 (54)	Driving straight, urban two-way road, clear noon
Udacity Straight1 (21)	Stop for a white SUV, urban one-way road, clear noon
Udacity Straight2 (22)	Driving Straight, urban one-way road, clear noon
Udacity Curve1 (15)	Turning left, urban two-way road, dusk
Kitti Curve1 (21)	Turning Right, urban one-way road, clear noon
Kitti Straight1 (21)	Driving Straight, urban one-way road, clear noon

TABLE III
MODEL PERFORMANCE (MSE) ON THE PROVIDED VALIDATION SET

Train & Val set	Model	Performance (MSE)	
		Steering	Acceleration
Carla100	CILR	0.0002	0.0382
Carla100	CILRS	0.0002	0.0343
Udacity	MT	0.0020	0.0079

Our manual selection of videos captures various scenarios under urban traffic, such as different weather conditions and different driving maneuvers. Further details regarding these 14 videos are summarized in Table II.

2) Autonomous Driving Model: In the assessment, we select three E2E autonomous driving models for testing: CILRS, CILR, and MT. Table III lists the dataset used for training and validation and the performance of models. Following common practice [13], [49] in autonomous driving research, we report mean square error (MSE) between actual and predicted values as a measure of model accuracy. For consistency, we compute MSE on steering in radians and acceleration in [0, 1] for all models on their provided validation sets.

CILRS and CILR are conditional imitation learning models, proposed by Codevilla et al. [8]. CILR uses the ResNet perception module as the backbone to extract information from RGB inputs. CILRS extends CILR architecture with an additional speed prediction head to incorporate speed-related features into the representation. They achieve outstanding performance in Carla's simulated urban traffic scenarios. Both models are trained using 10-hours of expert demonstrations and validated on a 2-hour subset (extracted from the Carla100 dataset). For both models, we use public-available pretrained weights.[2] Readers can refer to the original work [8] for training details (e.g., training objectives, hyperparameter settings). On the provided validation set, both models achieve 0.0002 MSE in steering and approximately 0.03 MSE in acceleration.

MT [50] utilizes RGB and optical flow images to learn both position and motion information. It employs two ResNet backbones for feature extraction from RGB and optical flow images, followed by a transformer encoder to distill knowledge from both. We use public-available pretrained weights provided by the original work.[3] During their training process (please refer to the original work [50] for training details), the first 90% of samples from the Udacity self-driving car challenge dataset are used for training, and the remaining 10% is used for validation. On the validation set, MT achieves 0.002 MSE in steering and 0.0079 in acceleration, surpassing the widely used steering angle prediction model, NVIDIA's DAVE2 [6], which only achieves 0.0449 MSE in steering based on our experiments.

For the experimentation, only RGB input undergoes perturbation, while other inputs (if there exists) remain unchanged. To maintain consistency and simplicity, we convert the output steering values from radians into degrees (°) and acceleration values into speed values (km/h) according to the data documentation.

3) Baseline: We test five baselines to evaluate the effectiveness of UniAda. To ensure a fair comparison, for all methods, we set a uniform maximum perturbation, we set the attack surface to the whole image, and we perform targeted adversarial attacks. We use the best hyperparameter values reported in their article for experiments.

Fast Gradient Sign Method (FGSM): Proposed by Goodfellow et al. [15], FGSM is a one-step gradient-based attack approach to craft image-specific adversarial perturbations. To facilitate a meaningful comparison with UniAda, we extend FGSM to perform targeted attacks on multiple car controls by employing an equal-weighted multiobjective loss (same loss as UniEqual) as the optimization criterion during the perturbation search.

DeepBillboard (DB): Introduced by Zhou et al. [10], DeepBillboard represents an iterative state-of-the-art attack method. It performs targeted attacks (intentionally misleads the model to steer left or right) and consistently misleads the model's steering predictions. The adversarial perturbation pattern generation process is intrinsically linked to the constraint applied to the steering angle. Due to this inherent constraint, unlike FGSM, it is challenging to extend DeepBillboard to multiple objectives. Consequently, we present the steering angle results exclusively.

DeepManeuver (DM) [21] is a state-of-the-art gradient-based adversarial test generation approach for autonomous vehicles. It generates a perturbation patch on the predefined attack surface, consistently misleading the model's steering prediction into the targeted direction (i.e., turn left or right). Similarly, we exclusively report steering results due to the design of its algorithm.

Perturbation Attack (PA) [28] assesses the impact of image-specific PAs and patch attacks on the 3D object detector. The former applies the attack to the entire image while the latter achieves the attack by applying a small patch to the input image. To enable a meaningful comparison with UniAda, we utilize the PA and an equal-weighted multiobjective loss (the same loss as UniEqual, enabling the specifications of attack directions) as the optimization criterion to induce misbehavior in car controls instead of 3D bounding boxes.

UniEqual: A variant of UniAda, sharing an identical universal multiobjective attack algorithm. However, UniEqual assigns equal weights to different objectives, omitting the use of AWS. Similar to UniAda, it is an iterative attack method designed to generate a universal multiobjective perturbation.

4) Evaluation Metrics: In this section, we introduce the evaluation metrics utilized in our experimentation. To assess the attack effectiveness of the generated examples, we adhere to established practices [10], [46] and employ two widely used metrics for the offline evaluation of ADS models: mean error (ME) [45] and success rate (SR) [46], as defined below for each objective i

$$\mathrm{ME}_i = \frac{1}{N}\sum_{n=1}^{N}[d_i \times (m_i(\mathbf{X}_{\text{ori}}^n + \tau) - m_i(\mathbf{X}_{\text{ori}}^n))] \tag{8}$$

$$\mathrm{SR}_i = \frac{1}{N}\sum_{n=1}^{N} I(d_i \times (m_i(\mathbf{X}_{\text{ori}}^n + \tau) - m_i(\mathbf{X}_{\text{ori}}^n)) > \delta_i). \tag{9}$$

The variables mentioned above are detailed in Section III. I is an indicator function that returns 0 or 1. Notably, the evaluation metrics are designed to capture the power of the attack. Larger metric values signify more effective attacks. In order to punish incorrect attack directions (i.e., situations when the attacker wishes the car to turn right but the attack algorithm misleads the car into turning left), the use of sign-invariant operators such as absolute or squared values in the metrics is avoided.

ME indicates the average strength of our attack on the input video. SR provides a hierarchy of performance by segmenting results based on different thresholds. More specifically, the SR measures the proportion of image frames in the video that have been successfully attacked. An attack is deemed successful if the discrepancy between the prediction made on the generated adversarial image and the original prediction surpasses a predetermined threshold. Thresholds for steering and acceleration are denoted as δ_S and δ_A, respectively. By testing various threshold values, we gain hierarchical insights into how the universal adversarial perturbation impacts a sequence of model predictions.

5) Hyperparameters: For the baseline methods, we use the best hyperparameters reported in their paper for the experimentation. Hyperparameters for UniAda are outlined and explained in Table I.

To ensure the generated perturbations remain imperceptible to humans, we set the maximum perturbation to $\epsilon = 2$ for simulated videos and $\epsilon = 5$ for real-world videos. This value is chosen to strike a balance between inducing misbehavior and maintaining the visual similarity between the original and perturbed images. In Fig. 4, we illustrate examples of adversarial images produced with perturbations of different ϵ values. Notably, for simulated

[2][Online]. Available: https://github.com/felipecode/coiltraine
[3][Online]. Available: https://github.com/chingisooinar/AI_self-driving-car

TABLE IV
ME RESULT TABLE: ME RESULTS FOR 7 SIMULATED VIDEOS OF ALL SIX TECHNIQUES, TESTED WITH CILRS AND CILR MODELS. STATISTICALLY SIGNIFICANT (TWO-SAMPLE T-TESTS) BEST-PERFORMING VALUES ARE STARRED

Videos	Metrics	CILRS						CILR					
		UniAda	DM	DB	PA	FGSM	UniEqual	UniAda	DM	DB	PA	FGSM	UniEqual
Carla Pedestrian	ME_S	39.9	**40.5**	32.0	32.6	1.40	39.5	**30.1**	23.9	12.7	22.5	0.20	30.0
	ME_A	22.9	–	–	16.1	15.6	**23.2**	**13.3** *	–	–	11.6	9.70	13.2
Carla White Car	ME_S	**25.6** *	18.1	20.5	16.1	1.40	24.6	**19.6**	10.4	9.79	11.1	0.90	19.5
	ME_A	**22.0** *	–	–	20.7	15.6	21.1	**15.4** *	–	–	14.6	12.9	14.8
Carla Black Car	ME_S	**18.6** *	6.25	5.47	13.5	0.70	9.63	**19.6** *	12.3	6.81	9.51	1.20	13.8
	ME_A	23.8	–	–	16.8	8.70	**23.9**	**25.4** *	–	–	18.6	6.00	21.3
Carla Gray Car	ME_S	**23.7** *	11.8	5.88	12.0	0.70	7.08	**14.8** *	2.21	12.0	11.8	1.20	13.3
	ME_A	**17.4** *	–	–	13.9	6.90	14.7	**18.2**	–	–	16.5	5.06	18.1
Carla Blue Car	ME_S	**28.6** *	24.8	26.1	14.5	1.40	22.3	**31.2** *	18.5	9.68	15.1	0.10	30.7
	ME_A	**22.6**	–	–	21.9	8.74	22.5	**24.3** *	–	–	21.2	8.28	23.8
Carla Red Light	ME_S	**41.9** *	38.9	35.4	21.2	0.70	41.2	**43.4** *	31.9	36.5	17.9	0.60	42.7
	ME_A	**26.3**	–	–	23.1	11.0	26.2	26.3	–	–	21.8	9.20	**26.4**
Carla Light-blue Car	ME_S	**25.9** *	17.5	15.0	6.37	0.70	11.2	20.4	19.6	**23.8** *	5.25	0.80	7.85
	ME_A	21.0	–	–	19.9	8.74	**22.0**	**14.0** *	–	–	0.91	0.92	4.16
Average	ME_S	**29.2** *	22.6	20.1	16.6	1.00	22.2	**25.6** *	17.0	15.9	13.3	0.71	22.6
	ME_A	**22.3**	–	–	18.9	10.8	21.9	**19.6** *	–	–	15.0	7.44	17.4

Last line indicates the average result over 7 videos. Best results are highlighted in bold. We approximate the floating steering angle to degrees $^\circ$ in ME_S and floating acceleration to km/h in ME_A for a more straightforward attack effect. - denotes unavailable results.

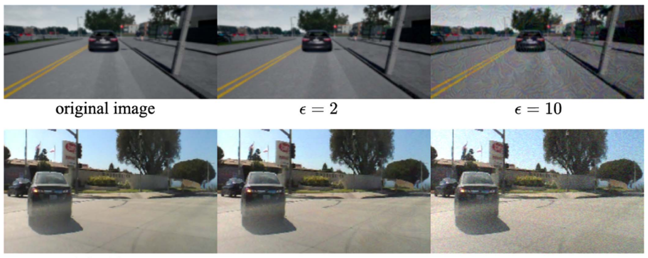

Fig. 4. Adversarial images with different ϵ values.

images with $\epsilon = 2$ and real-world images with $\epsilon = 5$, the perturbed image closely resembles the original. For higher ϵ values, the distinction between an adversarial image and its original counterpart becomes more evident.

Reproducibility: The trained model, framework, and data are available at https://github.com/UniAdaRepo/UniAda/.

C. Results and Analyses

In this section, we present our results and analyses to address the research questions. For all methods, the targeted attack directions are set to $d_S = 1$ (steering to the right) and $d_A = 1$ (acceleration). Positive/Negative error values indicate alignment/misalignment with the specified attack directions. All experiments are repeated five times and average results are reported.

Visualization of Generated Adversarial Images: We provide visualizations of example adversarial images generated by UniAda in Fig. 5 for both simulated and real-world data. The top/bottom row displays adversarial images generated from simulated/real-world dataset. Red/blue arrows represent the original/adversarial steering angle prediction. The bottom-right corner of each image indicates the acceleration error.

RQ1. UniAda Effectiveness: We evaluate the effectiveness of UniAda in comparison to four baselines: DM, PA, DB, and FGSM, using two evaluation metrics across 14 videos and 3 ADSs.

TABLE V
ME RESULT FOR 7 REAL-WORLD DRIVING VIDEOS OF ALL SIX TECHNIQUES WITH MT MODEL UNDER TEST. AVG. DENOTES THE AVERAGE RESULTS OVER 7 VIDEOS. STATISTICALLY SIGNIFICANT BEST-PERFORMING VALUES ARE STARRED

Videos	Metrics	MotionTransformer					
		UniAda	DM	DB	PA	FGSM	UniEqual
Dave Curve1	ME_S	**7.25** *	1.54	1.88	5.69	0.94	5.73
	ME_A	**14.3** *	–	–	5.21	2.54	3.97
Dave Straight1	ME_S	**1.01** *	-0.01	0.25	0.30	-0.08	0.63
	ME_A	**4.68** *	–	–	-5.35	-7.55	-2.33
Udacity Straight1	ME_S	**8.15**	7.47	7.31	8.11	4.51	7.99
	ME_A	**16.5** *	–	–	10.1	3.55	6.18
Udacity Straight2	ME_S	**2.75** *	-0.25	-0.01	0.73	0.14	1.57
	ME_A	**10.3** *	–	–	-4.65	-2.06	1.29
Udacity Curve1	ME_S	**1.97**	-3.09	-2.18	1.07	0.83	1.94
	ME_A	**12.2** *	–	–	3.00	1.99	-2.35
Kitti Curve1	ME_S	2.90	2.76	**3.14**	2.90	0.84	3.10
	ME_A	**7.20** *	–	–	1.41	0.34	-0.59
Kitti Straight1	ME_S	**0.74** *	-4.16	-2.45	0.09	0.54	0.63
	ME_A	**11.7** *	–	–	1.01	-1.65	1.75
Avg.	ME_S	**3.54** *	0.61	1.13	2.69	1.10	3.08
	ME_A	**11.0** *	–	–	1.53	-0.41	1.13

ME Results: Results for simulated and real-world data are presented in Tables IV and V. UniAda outperforms all four baselines in both steering angle and acceleration objectives in the average results for both simulated and real-world data. For simulated videos, under CILRS, UniAda causes an average steering error of approximately 29.2°, which is 6.6°, 9.1°, 12.6°, and 28.2° greater than DM, DB, PA, and FGSM, respectively. For acceleration, UniAda leads to an acceleration error of around 22.3 km/h, surpassing PA and FGSM by 3.4 and 11.5 km/h, respectively. For CILR, UniAda maintains its leading position for both objectives, causing an average steering error of 25.6°, with a maximum error of 43.4° under "Red Light." For acceleration, it reaches an average error of 19.6 km/h. Among the four baselines, for both models on average, DM demonstrates the best performance, while FGSM performs the worst. DB and PA secure middle positions. Delving into the performance of individual videos, we found that under the CILR "Light-blue Car" video, UniAda is slightly outperformed by DB in ME_S by around 3.4°. Similarly, in CILRS "Pedestrian," UniAda is slightly outperformed by DM by 0.6°. However, DB and DM can only target one objective

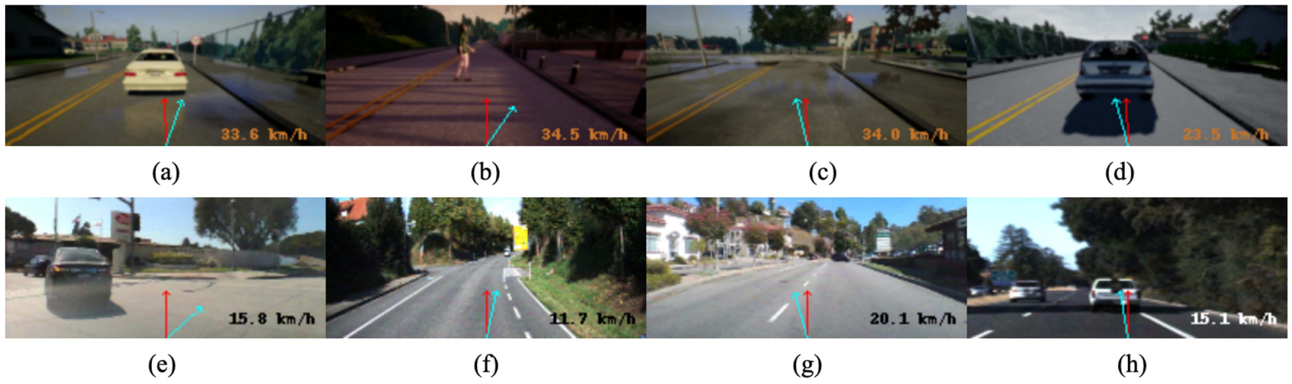

Fig. 5. Adversarial Images generated by UniAda: The first/second row is adversarial images generated from simulated/real-world videos. Blue/Red arrows indicate the adversarial/original steering angle predicted from the image. The bottom right shows the acceleration error. (a) Carla White Car, right. (b) Carla Pedestrian, right. (c) Carla Red Light, left. (d) Carla Light-blue Car, left. (e) Dave Curvel, right. (f) Kitti Curvel, right. (g) Dave Straightl, left. (h) Udacity Straightl, left.

at a time, making them less functional compared to UniAda. Furthermore, when considering other videos and the average result, UniAda consistently outperforms them by a significant margin.

For real-world videos, UniAda reaches the best performance on average, with 3.54° in ME_S and 11.0 km/h in ME_A, which is about triple ME_S of DB. Only in one case (“Kitti Curve1”) DB slightly outperforms UniAda by 0.24°. On average, PA performs the best among four baselines, but is 9.47 km/h and 0.85° smaller than UniAda. DM yields the worst performance, resulting in 0.61° in ME_S, which may be attributed to the fact that DM is designed for online testing within simulators.

Targeted Attack Effectiveness: From Table IV, all methods demonstrate positive error values, indicating their effectiveness in misleading CILRS and CILR in the intended direction. However, when attacking the MT model (see Table V), all methods except UniAda occasionally generate adversarial samples that mislead the model into unintended directions, thereby failing to accomplish targeted attacks. In addition, all techniques achieve a relatively low error compared to attacking CILRS and CILR. The weak performance in attacking the MT model may be attributed to the fact that, according to our settings, only the RGB input undergoes perturbations, whereas the MT model extracts features from both RGB and optical flow images. Out of 7 videos, FGSM induces MT to decelerate in 3 cases, and 2 cases for PA (i.e., negative ME_A, which contradicts the intended acceleration target). In teams of steering angle, FGSM incorrectly induces the MT to turn left instead of right as instructed in one case, while PA, DB, and DM do so in zero, three, and four instances, respectively. Notably, FGSM exhibits more successful targeted attacks than state-of-the-art DB and DM, while also achieving a larger average ME_S than DM does. One possible reason for DB could be attributed to its algorithm design, which updates the perturbation based on the absolute steering error, neglecting its sign. Meanwhile, DM is originally designed for online testing within a simulated environment, thereby lacking the ability to generalize effectively to real-world data. Moreover, both DM and DB apply a single, image-agnostic perturbation across the entire video, whereas FGSM and PA generate unique perturbations to each image (image-specific). This may potentially limit DB and DM’s efficacy in targeted attacks on models with complex architectures.

SR Results: The SR, indicating the percentage of adversarial images successfully exhibiting misleading behavior under a specified threshold, is assessed for different bounds. In Table VI, we select four different bounds for each objective to assess the hierarchy of performance, with $\delta_S = 3.5°, 14°, 21°, 28°$ for the steering angle and $\delta_A = 4.6, 13.8, 23.0, 32.2$ km/h for acceleration.

UniAda consistently outperforms four baseline methods across all thresholds and all models, except for the case when $\delta_S = 28°$ under MT model. For example, UniAda successfully attack CILRS model to produce a steering prediction error of at least 28° for 45.4% of tested images, compared to DM’s 38.9%, DB’s 34.3%, PA’s 15.1% and FGSM’s 0% at the same threshold. For MT model, UniAda can trigger errors on much more images than baselines for acceleration, with at most 43.5% more images than the second best (i.e., PA). FGSM fails to produce a successful attack under nearly half of the tested thresholds.

Result 1: UniAda consistently outperforms all four baseline methods in Mean Error on average for all ADS models. For Success Rate, UniAda demonstrates the best performance under 23 out of 24 cases on average. Compared with the state-of-the-art technique, DM, UniAda achieves a mean steering error improvement of 6.6° in CILRS, 8.6° in CILR, and 2.93° in MT model.

RQ2. AWS Effectiveness: To assess the effectiveness of the AWS, we introduce a variant of UniAda dubbed UniEqual, which utilizes the same universal multiobjective attack algorithm but employs equal-weighted objectives during the perturbation search process (i.e., UniEqual omits the use of AWS).

Observing the ME results in Tables IV and V, it is evident that UniAda consistently outperforms UniEqual in both steering and acceleration objectives on average across all ADS models. This indicates that AWS effectively enhances the attack performance in both objectives simultaneously. While in certain videos, UniEqual might achieve a slightly higher attack error in one objective, it becomes apparent that it is due to the lack of focus on the other objective. For instance, in the CILRS model videos “Black Car” and “Light-blue Car,” UniEqual produces

TABLE VI
MEAN SR RESULT AVERAGE OVER ALL TESTING VIDEOS FOR THREE ADSS: CILRS AND CILR RESULTS ARE AVERAGED OVER 7 CARLA100 VIDEOS. STATISTICALLY SIGNIFICANT BEST-PERFORMING VALUES ARE STARRED

Methods		CILRS				CILR				MT			
	δ_S	3.5	14	21	28	3.5	14	21	28	3.5	14	21	28
	δ_A	4.6	13.8	23.0	32.2	4.6	13.8	23.0	32.2	4.6	13.8	23.0	32.2
UniAda	SR_S	**96.3** *	**89.5** *	**76.5** *	**45.4** *	**90.5**	**75.3** *	**62.2** *	**45.1** *	**33.5**	**12.3**	**8.34**	5.34
	SR_A	**74.7** *	**67.4** *	**62.3** *	**49.2**	**67.8** *	**61.1** *	**55.7** *	**43.9** *	**83.7** *	**31.6** *	**2.72** *	0.00
DM	SR_S	87.6	67.2	52.5	38.9	83.4	57.3	33.5	19.6	27.9	11.2	8.32	5.34
	SR_A	–	–	–	–	–	–	–	–	–	–	–	–
DB	SR_S	83.3	48.8	45.6	34.3	66.1	41.7	29.2	20.2	28.2	11.2	8.32	6.02
	SR_A	–	–	–	–	–	–	–	–	–	–	–	–
PA	SR_S	86.2	54.6	24.3	15.1	82.0	38.6	18.6	11.0	30.7	11.9	7.68	5.38
	SR_A	67.4	60.8	52.3	41.2	55.2	48.0	41.7	28.7	40.2	9.56	0.68	0.00
FGSM	SR_S	3.99	0.00	0.00	0.00	1.32	0.00	0.00	0.00	9.89	3.67	0.79	0.00
	SR_A	43.6	35.4	28.8	18.3	37.2	27.5	21.6	11.4	16.8	0.00	0.00	0.00
UniEqual	SR_S	83.4	57.9	48.0	37.6	84.2	57.3	47.1	38.8	32.0	11.6	8.06	6.02
	SR_A	73.7	65.9	60.1	47.6	59.4	52.6	47.3	36.3	28.3	5.85	0.00	0.00

MT results are averaged over 7 real-world driving videos. We tested thresholds δ_S=3.5, 14, 21, 28 in degrees and δ_A=4.6, 13.8, 23.0, 32.2 in km/h.

TABLE VII
P-VALUE RESULTS OF ME UNDER CILRS, CILR AND MT: RESULTS ARE SHOWN IN BOLD IF THEY HAVE A *P*-VALUE ≤ 0.05, THE POSITIVE/NEGATIVE/EQUAL SIGNS INDICATE THAT UNIADA HAS A BETTER/WORSE/EQUAL PERFORMANCE VERSUS UNIEQUAL

Videos	CILRS		CILR		Videos	MotionTransformer	
	ME_S	ME_A	ME_S	ME_A		ME_S	ME_A
Carla Pedestrian	7.61E-01(+)	1.67E-01(-)	8.88E-01(+)	**3.95E-03(+)**	Dave Curve1	**2.58E-08(+)**	**1.18E-09(+)**
Carla White Car	**3.49E-02(+)**	**3.19E-02(+)**	8.05E-01(+)	**2.64E-02(+)**	Dave Straight1	**1.20E-02(+)**	**1.71E-03(+)**
Carla Black Car	**5.93E-03(+)**	7.85E-01(-)	**4.39E-02(+)**	**1.53E-02(+)**	Udacity Straight1	**1.28E-02(+)**	**1.27E-10(+)**
Carla Gray Car	**6.28E-07(+)**	**7.48E-07(+)**	**7.40E-05(+)**	9.25E-01(+)	Udacity Straight2	**2.29E-03(+)**	**1.43E-05(+)**
Carla Blue Car	**3.58E-02(+)**	5.38E-01(+)	**4.99E-05(+)**	**1.73E-08(+)**	Udacity Curve1	6.03E-01(+)	**4.08E-15(+)**
Carla Red Light	**4.37E-02(+)**	8.62E-01(+)	**2.97E-03(+)**	1.50E-01(-)	Kitti Curve1	1.75E-01(-)	**3.00E-04(+)**
Carla Light-blue Car	**4.28E-07(+)**	3.81E-01(-)	**2.63E-04(+)**	**1.05E-05(+)**	Kitti Straight1	**4.75E-04(+)**	**3.76E-12(+)**
Average	**2.72E-06(+)**	2.63E-01(+)	**8.89E-04(+)**	**1.43E-04(+)**	Average	**2.71E-05(+)**	**1.08E-09(+)**

23.9 and 22.0 km/h acceleration error, only 0.1 and 1.0 km/h higher than UniAda, respectively. However, UniAda showcases a steering error of 18.6° and 25.9°, which is significantly higher than UniEqual, with 8.97° and 14.7°, respectively.

For real-world driving videos, UniAda consistently outperforms UniEqual in all seven videos, with an average improvement of 9.87 km/h in acceleration error and 0.46° in steering error. We can see that AWS maintains the attack strength on steering objective while significantly improves the attack power on acceleration. Moreover, it is noteworthy that in 3 out of 7 cases, UniEqual performs attack in an incorrect direction for the acceleration objective (i.e., misalignment with the specified direction). In contrast, AWS optimizes UniAda to successfully attack all tested videos in the correct direction as specified. This is attributed to the AWS feature of balancing the training rate of both objectives by adapting weights, which is to ensure both objectives are thoroughly explored and collectively contribute to the perturbation that significantly affects both objectives simultaneously.

Similar conclusion applies to the SR results. As displayed in Table VI, UniAda outperforms UniEqual in all models across all tested thresholds, except for one instance: $\delta_S = 28°$ under MT model. In this case, perturbation generated by UniEqual can mislead slightly more images than UniAda with only 0.68% improvement. Nevertheless, with the same perturbation, UniAda can mislead 55.4%, 25.8%, 2.72% more images than UniEqual at $\delta_A = 4.6$, 13.8, 23.0 km/h, respectively.

Statistical Significance: We have conducted two-sample *t*-tests [51] to assess whether the performance differences between UniAda and UniEqual are statistically significant. Specifically, we treat all five runs of results as a sample group, with the null hypothesis stating that the means of two populations are the same. Each evaluation metric is examined for each driving model for both real-world and simulated driving data. Tables VII and VIII demonstrate the *p*-value results for ME and SR, respectively. Individual video results offer a detailed performance comparison for specific scenarios. To examine the statistical significance of overall performance difference under the simulated/real-world driving environment, we also present *p*-values for the average results (for each run, we compute the average results of seven videos, and then perform *t*-tests with five runs as a group). Results are shown in bold if they have a *p*-value ≤ 0.05, the positive/negative/equal signs indicate that UniAda has a better/worse/equal performance versus UniEqual. For ME *p*-value results, there are 30 out of 42 cases (exclude average results) UniAda statistically significantly outperforms UniEqual. Out of these, 8 cases come from CILRS, 10 cases for CILR, and 12 cases for MT. Moreover, we note that for the videos that UniEqual outperforms UniAda, the results are not

TABLE VIII
P-VALUE RESULTS OF SR UNDER CILRS, CILR, AND MT

	3.5	14	21	28
δ_S / δ_A	4.6	13.8	23.0	32.2
CILRS	**2.64E-04(+)**	**1.39E-04(+)**	**7.44E-05(+)**	**1.15E-03(+)**
	3.93E-02(+)	**4.09E-02(+)**	**2.32E-02(+)**	9.92E-02(+)
CILR	5.43E-02(+)	**5.37E-03(+)**	**2.98E-02(+)**	**4.59E-02(+)**
	1.37E-03(+)	**2.17E-03(+)**	**1.35E-03(+)**	**2.29E-02(+)**
MT	3.40E-01(+)	**9.23E-05(+)**	**5.56E-09(+)**	7.32E-02(-)
	3.72E-08(+)	**7.01E-08(+)**	**2.97E-11(+)**	NA(=)

"NA" denotes unavailable results due to the same performance of UniAda and UniEqual.

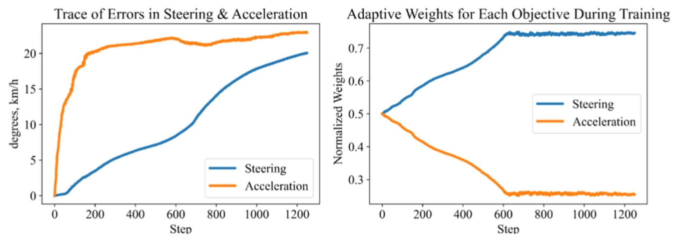

Fig. 6. Traces of ME (left) and AWS normalized weights (right) by UniAda attack in CILRS "Black Car."

TABLE IX
ABLATION STUDY ON CILRS AND MT MODELS, AVERAGED OVER TESTING VIDEOS

	CILRS			MT		
	L_{Steer}	L_{Acc}	UniAda	L_{Steer}	L_{Acc}	UniAda
ME$_S$	23.7	-	29.2	3.26	-	3.54
ME$_A$	-	19.5	23.0	-	9.82	11.0

significant (e.g., "Black Car," "Light-blue Car"). For SR p-value results, in 19 out of 23 cases, UniAda is significantly better than UniEqual.

To illustrate how AWS functions, we demonstrate an example plot (CILRS "Black Car," Fig. 6) of error and weight traces against the perturbation searching process. Each objective weight begins at 0.5 with zero errors. During the initial stages, the attack error in acceleration experiences a substantial improvement, whereas the steering perturbation triggers minimal errors. This discrepancy may be attributed to the higher training rate allocated to acceleration. Consequently, the AWS assigns more weight to the steering objective, driving up the steering error while stabilizing the acceleration error after around 200 steps. The objective weights reach a steady state after approximately 600 steps. At the later stage, although the weights are mainly allocated to steering, the acceleration error still shows a further increase, which implies the perturbation pattern may contain shared information that also triggers errors in acceleration.

Result 2: AWS assists UniAda to improve the performance in both objectives simultaneously in most cases. On average, AWS improves ME_S by 7.0°, 3.0°, 0.46°, and ME_A by 0.4 km/h, 2.2 km/h, 9.87 km/h, for CILRS, CILR, and MT, respectively.

RQ3. Multi-objective Attack Effectiveness: To assess the effectiveness of the multiobjective attack, we conduct experiments involving two single-objective attacks and compare their results with UniAda for CILRS and MT models. The single-objective attacks utilize the same algorithm as UniAda, differing solely in the loss function employed. We report the average results over the corresponding testing videos, summarized in Table IX, with attack directions set at $d_S = 1$ and $d_A = 1$.

On average, UniAda consistently outperforms both single-objective attacks. Specifically, in terms of steering, UniAda induces a steering angle misdirection of 5.5° more than the single-objective attack focused on steering (L_{Steer}). Similarly,

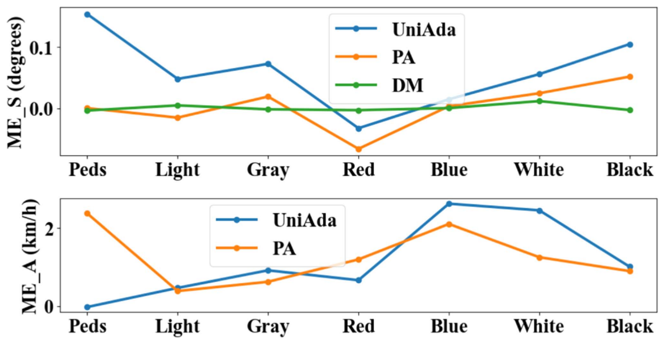

Fig. 7. Transferability of Adversarial attacks: ME results in Steering (top) and Acceleration (bottom) of UniAda, DM, PA. Note that DM results for ME_A is not available.

for acceleration, UniAda generates an acceleration error around 3.5 km/h higher compared to the single-objective acceleration attack (L_{Acc}). Similar results for MT model, where single-objective attack is 0.28° and 1.18 km/h less than that of UniAda. This outcome suggests that the perturbation patterns sought for steering and acceleration objectives complement each other, showcasing the advantage of multiobjective attacks.

Result 3: Compared to single objective attacks, the multi-objective attack boosts the performance for both objectives simultaneously for both simulated and real-world videos.

VII. DISCUSSION

A. Transferability of Adversarial Attacks

In this section, we analyze the transferability of the generated adversarial perturbation on different autonomous driving models. Specifically, we explore the attack effectiveness of the perturbation generated with CILRS on CILR model for each video in Carla100 dataset. We compare the ME results of UniAda with its two closest competitors, DM and PA, shown in Fig. 7.

As shown in the plot, for ME_S, the perturbation generated by UniAda with CILRS model under test can cause the highest error when attack CILR model in most cases. Only in the video "Red Light," DM produces a better performance. For ME_A, UniAda outperforms PA in 5 out of 7 cases. Only in "Pedestrian," PA outperforms UniAda by a relatively large margin. However, we can see that all three techniques exhibit weak transferability, a phenomenon frequently observed under the white-box setting owing to the attack's specificity to the internal structure of the model [52]. Possible directions to address this issue include using advanced gradient calculation [53], or applying various input transformations [54]. We leave the improvement of transferable attacks for further work.

B. Limitations

UniAda is a gradient-based approach that requires white-box access, leading to less generalizability on DNNs with restricted access or knowledge. Moreover, our findings on real-world videos is limited by one victim DNN, MT. The choice of MT is due to its complex architecture (i.e., incorporates RGB and optical flow images to learn both position and motion information) and its competitive performance (e.g., surpassing CNN-based DAVE2 model). Our findings are also limited to the parameter space we explored, capturing key factors influencing the techniques' performance but is still limited in scope.

C. Threats to Validity

Internal Validity: The threats to internal validity can be attributed to the implementation quality of baselines and reproducibility of our method. To mitigate this, we open-source our implementation to facilitate checking and reproducibility.

External Validity: The first threat to external validity to our experimental conclusions is our selected datasets (i.e., 7 simulated videos from Carla100 dataset, 7 real-world videos from Dave, Kitti, and Udacity datasets) and DNN models (i.e., CILRS, CILR, MT). We tried to alleviate this threat as follows: 1) the selected datasets contain both simulated and real-world driving scenarios. They are also widely used in the previous research [8], [10], [12]; and 2) the three autonomous driving models have achieved competitive performance in the field. They are constructed with different architectures and different numbers of layers. For example, CILR and CILRS use ResNet only to learn RGB data information, while MT is composed of more complex architecture that incorporates optical flow with RGB images to capture both position and motion information. Therefore, our experimental conclusions should generally hold with other driving datasets and models.

The second external validity concern is the selection of baselines. To mitigate this, we compare UniAda with DeepManeuver [21], PA [28], DeepBillboard [10], and FGSM [15] that are either state-of-the-art (e.g., DeepManeuver, published in 2023) or representative (e.g., DeepBillboard, FGSM) in this field.

Construct Validity: Construct validity concerns whether the chosen evaluation metrics accurately capture the intended effect. In our study, our goal is to design an attack technique that generates image-agnostic perturbations capable of inducing errors in the model under test. We mitigate the threat by examining the attack effectiveness on two metrics, ME and SR, which are widely used in the literature [10], [12], [45], [46]. The ME can reflect the average attack strength, however, it might be biased by some outliers and cannot represent the performance of most input images. To counter this threat, we also measure the SR, which manifests the universality of the attack (i.e., whether the generated perturbation can affect most input images) by examining the percentage of images that are successfully attacked under different thresholds.

VIII. Conclusion

In this article, we examine the reliability and security problems raised by adversarial attacks on three E2E ADSs. Our proposed method, UniAda, conducts novel and comprehensive testing by addressing three main limitations from existing literature: limited vehicle controls testing, constrained testing scenarios, and image-specific or unnatural perturbations. We alleviate them accordingly by 1) performing testing on steering and acceleration controls simultaneously through the multi-objective attack with AWS, 2) examining both simulated and real-world driving scenarios in urban traffic, and 3) generating image-agnostic (i.e., universal) and human-eye-imperceptible perturbation through joint optimization. The effectiveness of UniAda is compared with four baselines in ME andSR metrics. Compared with its closest competitor, DeepManeuver, UniAda achieves an improvement of 6.6°, 8.6°, and 2.9° in steering error on CILRS, CILR, and MT ADSs, respectively.

References

[1] M. HUTSON, "Watch just a few self-driving cars stop traffic jams," 2018. [Online]. Available: https://www.science.org/content/article/watch-just-few-self-driving-cars-stop-traffic-jams

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[3] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1364–1384, Apr. 2022.

[4] E. D. Dickmanns, "The development of machine vision for road vehicles in the last decade," in *Proc. Intell. Veh. Symp.*, 2002, pp. 268–281.

[5] J. Leonard et al., "A perception-driven autonomous urban vehicle," *J. Field Robot.*, vol. 25, no. 10, pp. 727–774, 2008.

[6] M. Bojarski et al., "End to end learning for self-driving cars," 2016, arXiv:1604.07316.

[7] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4693–4700.

[8] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9329–9338.

[9] Udacity, "Using deep learning to predict steering angles," 2016. [Online]. Available: https://medium.com/udacity/challenge-2-using-deep-learning-to-predict-steering-angles-f42004a36ff3

[10] H. Zhou et al., "DeepBillboard: Systematic physical-world testing of autonomous driving systems," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng.*, 2020, pp. 347–358.

[11] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems," in *Proc. 33rd IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2018, pp. 132–142.

[12] Z. Kong, J. Guo, A. Li, and C. Liu, "PhysGAN: Generating physical-world-resilient adversarial examples for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14254–14263.

[13] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 1–18.

[14] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *stat*, vol. 1050, p. 20, 2015.

[16] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learn. Representations*, 2016.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. London, U.K.: Chapman and Hall/CRC, 2018, pp. 99–112.

[18] F. Tramér, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[19] Z. Li, M. Pan, T. Zhang, and X. Li, "Testing DNN-based autonomous driving systems under critical environmental conditions," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 6471–6482.

[20] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 303–314.

[21] M. von Stein, D. Shriver, and S. Elbaum, "DeepManeuver: Adversarial test generation for trajectory manipulation of autonomous vehicles," *IEEE Trans. Softw. Eng.*, vol. 49, no. 10, pp. 4496–4509, Oct. 2023.

[22] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng.*, 2020, pp. 359–371.

[23] C. Zhang, P. Benz, C. Lin, A. Karjauv, J. Wu, and I. S. Kweon, "A survey on universal adversarial attack," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, doi: 10.24963/ijcai.2021/635.

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1765–1773.

[25] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 794–803.

[26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[27] Sully-Chen, "Autopilot-tensorflow," 2016. [Online]. Available: https://github.com/SullyChen/Autopilot-TensorFlow

[28] J. Zhang, Y. Lou, J. Wang, K. Wu, K. Lu, and X. Jia, "Evaluating adversarial attacks on driving safety in vision-based autonomous vehicles," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3443–3456, Mar. 2022.

[29] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2021.

[30] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "TransFuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 12878–12895, Nov. 2023.

[31] S. Casas, A. Sadat, and R. Urtasun, "MP3: A. unified model to map, perceive, predict and plan," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14403–14412.

[32] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," *Adv. Neural Inf. Process. Syst.*, vol. 1, pp. 305–313, 1988.

[33] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin, and Q.-L. Han, "Deep learning-based autonomous driving systems: A survey of attacks and defenses," *IEEE Trans. Ind. Inform.*, vol. 17, no. 12, pp. 7897–7912, Dec. 2021.

[34] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. 2017 ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.

[35] M. Liu, H. Zhang, Z. Liu, and N. Zhao, "Attacking spectrum sensing with adversarial deep learning in cognitive radio-enabled Internet of Things," *IEEE Trans. Rel.*, vol. 72, no. 2, pp. 431–444, Jun. 2023.

[36] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, "Adversarial attacks in modulation recognition with convolutional neural networks," *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 389–401, Mar. 2021.

[37] P. Qi, T. Jiang, L. Wang, X. Yuan, and Z. Li, "Detection tolerant black-box adversarial attack against automatic modulation classification with deep learning," *IEEE Trans. Rel.*, vol. 71, no. 2, pp. 674–686, Jun. 2022.

[38] T. Woodlief, S. Elbaum, and K. Sullivan, "Semantic image fuzzing of AI perception systems," in *Proc. 44th Int. Conf. Softw. Eng.*, 2022, pp. 1958–1969.

[39] S. Pavlitskaya, S. Ünver, and J. M. Zöllner, "Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–8.

[40] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[41] F. Codevilla, A. M. Lopez, V. Koltun, and A. Dosovitskiy, "On offline evaluation of vision-based driving models," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 236–251.

[42] F. U. Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *Proc. IEEE 13th Int. Conf. Softw. Testing Validation Verification*, 2020, pp. 85–95.

[43] H. Liu and H. B. K. Tan, "Covering code behavior on input validation in functional testing," *Inf. Softw. Technol.*, vol. 51, no. 2, pp. 546–553, 2009.

[44] S. Nidhra and J. Dondeti, "Black box and white box testing techniques-a literature review," *Int. J. Embedded Syst. Appl.*, vol. 2, no. 2, pp. 29–50, 2012.

[45] H. Wu, S. Yunas, S. Rowlands, W. Ruan, and J. Wahlström, "Adversarial driving: Attacking end-to-end autonomous driving," in *Proc. IEEE Intell. Veh. Symp.*, 2023, pp. 1–7.

[46] Y. Deng, X. Zheng, T. Zhang, C. Chen, G. Lou, and M. Kim, "An analysis of adversarial attacks and defenses on autonomous driving models," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2020, pp. 1–10.

[47] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[48] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?," 2019, arXiv:1908.01878.

[49] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng.*, 2019, pp. 1039–1049.

[50] C. Oinar and E. Kim, "Self-driving car steering angle prediction: Let transformer be a car again," 2022, arXiv:2204.12748.

[51] W. S. Gosset, "The probable error of a mean," *Biometrika*, vol. 6, pp. 1–25, 1908.

[52] X. Wang and K. He, "Enhancing the transferability of adversarial attacks through variance tuning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1924–1933.

[53] Y. Dong et al., "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.

[54] C. Xie et al., "Improving transferability of adversarial examples with input diversity," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2730–2739.