# Benchmarking Image Perturbations for Testing Automated Driving Assistance Systems

Stefano Carlo Lambertenghi
*Technical University of Munich, fortiss*
Munich, Germany
stefanocarlo.lambertenghi@tum.de,
lambertenghi@fortiss.org

Hannes Leonhard
*Technical University of Munich*
Munich, Germany
hannes.leonhard@tum.de

Andrea Stocco
*Technical University of Munich, fortiss*
Munich, Germany
andrea.stocco@tum.de,
stocco@fortiss.org

***Abstract*—Advanced Driver Assistance Systems (ADAS) based on deep neural networks (DNNs) are widely used in autonomous vehicles for critical perception tasks such as object detection, semantic segmentation, and lane recognition. However, these systems are highly sensitive to input variations, such as noise and changes in lighting, which can compromise their effectiveness and potentially lead to safety-critical failures.**

**This study offers a comprehensive empirical evaluation of image perturbations, techniques commonly used to assess the robustness of DNNs, to validate and improve the robustness and generalization of ADAS perception systems. We first conducted a systematic review of the literature, identifying 38 categories of perturbations. Next, we evaluated their effectiveness in revealing failures in two different ADAS, both at the component and at the system level. Finally, we explored the use of perturbation-based data augmentation and continuous learning strategies to improve ADAS adaptation to new operational design domains. Our results demonstrate that all categories of image perturbations successfully expose robustness issues in ADAS and that the use of dataset augmentation and continuous learning significantly improves ADAS performance in novel, unseen environments.**

## I. Introduction

Advanced Driver Assistance Systems (ADAS) heavily rely on perception systems (e.g., cameras, LiDAR, and other sensors) to perceive complex, dynamic environments in real-time. These systems adopt deep neural networks (DNNs) for interpreting sensor data to assist tasks such as object detection, image classification, semantic segmentation, and regression, to enable accurate real-time driving functions [1], [2], [3], [4].

Despite their effectiveness in driving image understanding, these systems are expected to operate reliably across a large group of domain environments and operational domains. However, it is still infeasible to collect all representative scenarios during data collection and training campaigns. Thus, after deployment and in-field operation, the ADAS is likely to encounter inputs that significantly differ from the training data. Particularly, DNN-based ADAS perception systems are highly sensitive to input variations, such as lighting, environmental changes, noise, changes in lighting, or minor shifts in perspective [3]. These factors can lead to significant prediction errors [5], [6], misclassifications, or inaccurate segmentations. Such errors can propagate to the vehicle's decision-making modules, potentially resulting in safety-critical failures.

In literature, synthetic image perturbations have been proposed and utilised to assess and enhance the robustness of DNNs [7], [8], [9], [10], [11], [12]. Image perturbations introduce controlled distortions to input images (e.g., by reducing the brightness or by blackening certain pixels), and have been used to simulate out-of-distribution conditions that challenge the robustness of the DNN to slight input variations [7], [8], [13], [14]. Additionally, image perturbations have been used for robustness/adversarial training, e.g., as a data augmentation strategy to enrich the training dataset with perturbed versions of the original, unperturbed images. This increases the diversity of the training set and helps the DNN to be invariant to various types of distortions, thereby improving its robustness.

In the context of ADAS testing, image perturbations have been employed by solutions such as DeepXplore [15], DeepTest [16] and DeepBillboard [17], using input transformations such as lighting changes and occlusions, real-world inspired synthetic perturbations like rain and fog, or synthetic adversarial billboards. These works target *offline* ADAS testing of perception systems, which has been shown to be inadequate at revealing system-level failures [18], [19], [20]. Among the *online* approaches for system-level ADAS testing, most research has focused on test generation to assess generalizability [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], while fewer studies have addressed robustness testing at the system level. Some work proposed real-time adversarial attacks [13], [35], [36], state-aware billboards [37], or generic sets of perturbations for runtime anomaly detection [14], [38]. Although these solutions have shown great potential in exposing many failures, a comprehensive empirical evaluation of different image perturbations is still missing in the literature. Additionally, an in-depth analysis of their effectiveness, particularly in relation to their latency when used at a system level and their tuning in terms of intensity, remains an open area of investigation.

To address this, in this paper, we provide the most comprehensive evaluation of image perturbations, benchmarking numerous perturbation techniques from the literature, across two popular ADAS tasks with increasing complexity. We first reviewed the literature on existing DNN image perturbations, identifying the most commonly used techniques for robustness evaluation. Then, we selected perturbations based on their

feasibility and relevance to autonomous driving scenarios, retaining 32 types. To ensure a more thorough assessment, we evaluated the perturbations according to different intensity levels, enabling fine-grained control over their severity while ensuring that the original image semantics are preserved.

We systematically evaluate the performance of these perturbations in terms of their ability to generate robustness failure inputs in two ADAS, considering factors such as efficiency and effectiveness. Our findings confirm that all image perturbations expose ADAS failures, at different levels of severity, both at a component and a system level. Moreover, despite some of these image perturbations reflecting specific corner-case operational conditions (e.g., extreme lighting or camera occlusion), many do not represent naturalistic perturbations (i.e., weather effects) that can occur during real-world driving. Thus, we have investigated the usefulness of existing common image perturbations for domain adaptation and generalizability during robustness retraining. Our findings reveal that adversarial retraining and continuous learning with common image perturbations allow ADAS to adapt to new environmental and naturalistic perturbations, thereby enhancing their robustness in real-world scenarios.

This paper makes the following main contributions:

- An empirical study comparing 32 image perturbations on two ADAS tasks, both at a component and at a system level. Our study identifies perturbation categories and intensities that are most effective for robustness failure exposure and retraining for domain adaptation.
- A library for ADAS robustness testing, called PerturbationDrive, which integrates several dozens of image perturbations for both offline and online robustness ADAS testing. PerturbationDrive can be used as a standalone library for offline testing, providing perturbations that can be applied to input images. It also offers APIs that integrate seamlessly into different driving simulators without requiring modifications to existing infrastructures, and it is designed to be highly modular and extensible. To encourage open research, we made our library and experimental data available [39].

## II. DNN Image Perturbations

### A. Literature Review

We performed a systematic review of papers mined from Scopus [40] and arXiv [41]. Scopus was selected for its comprehensive archive of peer-reviewed articles, while arXiv allowed us to include grey literature and preprints in our review. These platforms helped us systematically search for papers that mentioned the keywords "DNN", "image perturbation" and "ADAS" within their abstract. This search found 2,943 papers, of which 1,814 in Scopus and 1,129 in arXiv. The search results from Scopus and arXiv were each ranked based on the number of citations and the publication year, with more recent papers prioritized, producing a list of ranked studies, which we reviewed starting from the highest-ranked paper. For each source, we identified all proposed image perturbations and added them to a catalogue of perturbations. If the paper referenced any additional sources related to perturbations, these were also ranked similarly and added to the study list for further review. We continued this process, moving through the ranked studies until the catalogue of perturbations remained unchanged for ten consecutive studies.

### B. List of DNN Image Perturbations

Table I presents the list of 38 perturbations obtained with our review of the literature, and their sources. We evaluated perturbations based on visual similarities or computational techniques and assigned them to eight main categories: noise, blur, focus, weather-related changes, affine transformations, graphic patterns, colour and tone adjustments, and generative-based. In the following, we briefly describe each category.

*1) Noise Perturbations (A):* This category includes random variations in pixel values or image graininess, typically resulting from electronic noise. This category includes: (A-I-II) Gaussian noise and Poisson noise, which consist in the addition of statistical noise to an image, using the probability density functions of the Normal distribution or the Poisson distribution, respectively. (A-III) Impulse noise, which consists of random, sharp and sudden disturbances, taking the form of scattered bright or dark pixels. (A-IV) JPEG artifacts, which perturbs an image in the same manner as JPEG compression artifacts would. (A-V) Speckle Noise, which adds granular noise textures to the image.

*2) Blur and Focus Perturbations (B):* These perturbations cause a reduction in image sharpness or clarity, often stemming from improper camera settings. A key characteristic of these perturbations is the use of kernels, which calculate the perturbation by averaging or smoothing pixel values in specific areas of the image. Adjustments to the size of the kernel matrix or changes to its values can modify the intensity of these perturbations. We have identified the following types: (B-I) Defocus blur, which simulates the effects of the camera lens being out of focus via disc-shaped kernels. (B-II) Motion blur, which mimics the streaking effects caused by moving objects. (B-III) Zoom blur, which simulates a radial blur that emanates from a central point of the image. (B-IV) Gaussian blur, which blurs the image by applying the Gaussian function on the image. (B-V) Low-pass filter, which calculates the average of each pixel to its neighbours.

*3) Weather Perturbations (C):* These perturbations simulate weather conditions such as snow, rain, or fog. Such conditions are influenced by specific times of the day or seasons; for instance, images taken at sunrise or sunset might exhibit enhanced brightness or contrast, while winter could naturally bring about snowfall. This category includes: (C-I) Frosted glass, which simulates the effects of frosting on a camera lens. (C-II) Snow, that simulates the presence of snow crystals. (C-III) Fog, that reduces the image's contrast and saturation to simulate the presence of fog. (C-IV) Brightness, that changes the image's brightness intensity, to simulate changes of environment lighting. (C-V) Contrast, that increases the difference of luminance on the image.

TABLE I: DNN image perturbation types identified in this study and their sources.

| **Noise Perturbations (A)** | **Blur Perturbations (B)** | **Focus Perturbations (C)** | **Weather Perturbations (D)** | **Affine Transformations (E)** | **Graphic Patterns (F)** | **Color/Tone Adjustments(G)** | **Generative-based (H)** |
|---|---|---|---|---|---|---|---|
| Gaussian Noise [5], [6], [7], [9], [10], [11], [14], [38], [42]<br>Poisson Noise [7], [9], [14], [43]<br>Impulse Noise [7], [9], [14], [43], [42]<br>JPEG [5], [6], [7], [9], [10], [14], [38], [42]<br>Speckle Noise [9], [11], [14] | Defocus Blur [7], [9], [14], [42]<br>Motion Blur [7], [9], [38], [43], [42]<br>Zoom Blur [7], [9], [38], [42]<br>Gaussian Blur [14], [16], [38]<br>Low Pass Filter [6] | Frosted Glass [7], [9], [38], [42]<br>Snow [7], [38], [42]<br>Fog [7], [9], [14], [38], [42]<br>Brightness [7], [14], [15], [16], [43], [42], [44], [45]<br>Contrast [5], [6], [7], [9], [12], [14], [16], [42], [44], [45] | Elastic [7], [9], [42]<br>Pixelate [7], [9], [42]<br>Sample Pairing [8], [14], [44]<br>Sharpen [12], [44], [45] | Shear Mapping [8], [12], [16], [43], [44], [45]<br>Scale [10], [12], [16]<br>Translate [8], [10], [16], [44], [45]<br>Rotate [6], [8], [10], [12], [16], [44], [45]<br>Reflection [12] | Splatter [14], [43]<br>Dotted Lines [43]<br>ZigZag [43]<br>Canny Edges [43]<br>Cutout [8], [15], [44] | False color [6], [45]<br>Phase scrambling[6]<br>Histogram equalization [6], [12], [44], [45]<br>White balance [12]<br>Greyscale [46]<br>Saturation [14], [42]<br>Posterize [8], [44] | Cycle-consistent [47], [48], [49]<br>Style-transfer [35], [42], [46] |

*4) Distortion Perturbations (D):* Distortion perturbations randomly displace or overlap image pixels, leading to distorted figures and shapes in the visuals. We identify the following perturbations: (D-I) Elastic moves each image's pixel by a random offset derived from a Gaussian distribution. (D-II) Pixelate divides the image in a set of pixel regions and sets the average pixel value of the region to all pixels in it. (D-III) Sample pairing randomly samples two regions of the image and blends them together with a varying alpha value. (D-IV) Sharpen removes blurring by using the sharpen kernel.

*5) Affine Transformations (E):* These transformations preserve the collinearity and parallelism of lines within the image, meaning that straight lines remain straight, and sets of parallel lines remain parallel after the transformation. However, the actual distances between points and the angles between lines can change. This allows for transformations such as rotation, scaling, translation, and shearing, fundamentally altering the image's appearance while maintaining a level of geometric consistency. These include: (E-I) Shear mapping shifts each point in an image horizontally. The shift's direction and magnitude are based on each point's perpendicular distance from a reference line parallel to the shift direction, resulting in a slanted or skewed appearance of the image. (E-II) Scale increases or decreases the size of the image by a certain factor. (E-III) Translate moves all the pixels of an image in a certain direction. (E-IV) Rotate rotates the image by a certain angle in the Euclidean space. (E-V) Reflection, creates a mirror effect by appending a flipped version of the image at a certain height.

*6) Graphic Patterns (F):* This type of perturbation involves inserting repetitive graphics and patterns randomly across the image. A pattern in this context is defined as a specific shape, like a dot or a rectangle, that is systematically repeated throughout the image. This category includes: (F-I) Splatter randomly adds black patches of varying size to the image. (F-II) Dotted lines randomly adds straight dotted lines to the image. (F-III) ZigZag randomly adds black zig-zagging lines to the image. (F-IV) Canny edge filter applies canny edge detection to highlight images and lay them over the images. (F-V) Cutout inserts black rectangular shapes on the image.

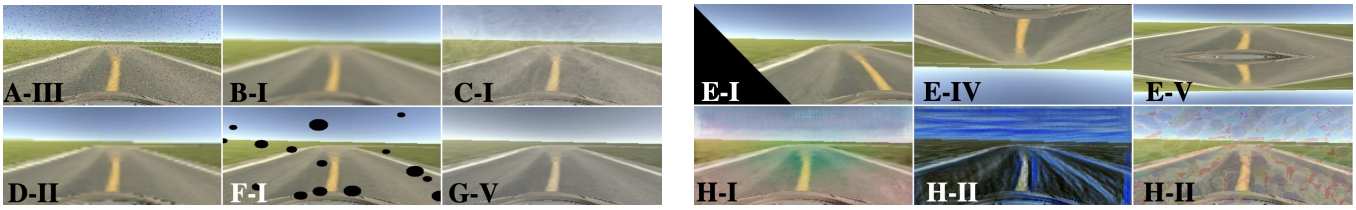
*7) Color and Tone Adjustments (G):* These perturbations modify the color and tone characteristics of the image by averaging, increasing, or decreasing specific channels. We have identified the following perturbations: (G-I) False color filter swaps color channels, inverts color channels, or averages color channels with each other. (G-II) Phase scrambling scrambles image channels using the Fast Fourier Transform. (G-III) Histogram equalization enhances the image contrast by spreading the pixel intensities using the image histogram. (G-IV) White balance globally adjusts the intensity of colors to adjust white portions of the image. (G-V) Greyscale filter converts the image to greyscale. (G-VI) Saturation increases or decreases the saturation of the image by changing the S channel of the image in HSV (Hue, Saturation, Lightness) representation. (G-VII) Posterize reduces the number of distinct colors by quantizing the color channels.

*8) Generative-based Perturbations (H):* The perturbations discussed in this section, use generative models to modify images, with the goal of changing the look of the input domain into the look of another alternative domain. In this category, we consider: (H-I) cycle-consistent models enable the generation of images across two image distributions representing different domains. (H-II) Style-transfer models apply artistic styles to images, using pre-trained generative models.

### *C. Image Perturbations Validity*

We manually analyzed each perturbation type with the aim to keep those that generate images that maintain the semantic between the original and the augmented image and that represent valid driving images (e.g., Figure 1(a)). We implemented these checks to make sure to use perturbations that do not drastically change the image content (i.e., making the road no longer visible, or an image in which the road is flipped horizontally, see Figure 1(b)), thereby exposing failures that are not relevant.

In particular, we filtered out perturbations such as some types of affine transformations, rotations, shear mapping, reflection, and generative based. Affine transformations can distort the image to the extent that generating a valid driving command, even for a human, becomes impossible. For example, shear mapping (E-I) skews and distorts the image,




(a) Valid. (b) Invalid.

Fig. 1: Valid and invalid perturbation types.

pushing parts outside the frame and potentially altering the ADAS's driving decisions. Rotations (E-IV) can shift images by an angle $\theta$, with larger angles (e.g., $\theta = 180°$) inverting the image, while reflections (E-V) duplicate content, both of which may confuse DNNs and mislead the ADAS. Similarly, generative-based perturbations (H) were excluded. CycleGAN (H-I) relies on the input domain matching the source domain of its training dataset, making it unsuitable for general driving scenarios, while style-transfer (H-II) often introduces unrealistic alterations, such as exaggerated textures and colour shifts, distorting the visual content to an unrealistic degree (Figure 1). For the segmentation task, we excluded the entire Affine Transformations (E) group because validating the model's performance would require distorting the pixel-level ground truth classes to match the transformations, which is not feasible for accurate evaluation.

To obtain the five intensity levels for the included perturbations, we resort to a visual assessment. Specifically, we incrementally applied each perturbation until we could no longer understand the depicted scene. The intensity level right before this threshold was considered as the maximum intensity. Then, we discretized the intensity range into five uniform steps.

### D. Implementation

To support our experimental evaluation, we develop an extensible Python library called PerturbationDrive, which is available [39]. It systematically enables the application of a large variety of image perturbations for conducting both model-level and system-level robustness testing of ADAS. Architecturally, PerturbationDrive consists of three main components: an image perturbation module, a simulator interface (only for the case of system-level testing), and a benchmarking controller. The image perturbation component implements all perturbations described in Section II and applies them to a given input image. It also supports the addition of new perturbations by extending an abstract interface.

For online testing, the framework integrates seamlessly with driving simulators. The initial release supports ADAS models developed in TensorFlow/Keras and is compatible with two Unity-based driving simulators: the Udacity Simulator [50] and the Sdsandbox Donkey Car™ simulator [51]. It enables the generation of different road layouts, represented as a series of waypoints in the 3D space, and allows the application of perturbations in real time.

Finally, the benchmarking controller manages the testing process. For offline testing, it applies perturbations to ADAS input images and compares the ADAS responses with either ground-truth values or its output on unperturbed images. During online testing, the framework logs various metrics such as the ADAS actions, perturbed and unperturbed images, and vehicle speed to determine if the system successfully drives the scenario or encounters a failure.

## III. Empirical Study

### A. Research Questions

**$RQ_1$ (effectiveness):** *Which types of image perturbations are more effective in inducing robustness failures in ADAS?*

The first research question investigates how different types of image perturbations, applied at varying levels of intensity, impact the reliability of ADAS.

**$RQ_2$ (generalization):** *How effective are common image perturbations in enhancing the generalization of ADAS to more naturalistic perturbations?*

The second research question investigates how effectively common perturbations can improve the generalization ability of ADAS. By introducing these perturbations during training or testing, we aim to determine whether the models become more resilient to real-world environmental scenario changes, such as weather conditions.

### B. Objects of Study

We consider NHTSA [52] Level 2 ADAS that perform vision-based perception tasks, i.e., from data gathered by camera sensors of a vehicle. Despite the adoption of Level 2 ADAS in many commercial vehicles, their reliability remains a concern, as evidenced by numerous recent crash reports [53]. Although Level 3 and 4 ADAS have been proposed [54], their real-world deployment remains highly constrained. Consequently, addressing the limitations of Level 2 systems is crucial for advancing to higher levels of autonomy.

We focus on two specific ADAS applications: a system for semantic segmentation and another designed for lane-keeping and adaptive cruise control (LK/ACC).

*1) Semantic Segmentation:* SegFormer [55] is a vision transformer-based model designed for semantic segmentation, where each pixel in an input image is classified into one of several object classes. The model employs a hierarchical transformer architecture for feature extraction and a lightweight

multi-level feature aggregation network to generate segmentation maps with both fine detail and global context. Unlike traditional convolutional models, SegFormer omits positional encodings, enhancing its efficiency and scalability for real-time applications, including ADAS. Trained on large-scale datasets like Cityscapes [56], SegFormer has demonstrated competitive performance in segmenting complex driving environments, which made it a reference model for ADAS testing studies [57], [58], [59], [60], [61].

*2) LK/ACC:* DAVE-2 is a convolutional neural network developed for multi-output regression tasks based on imitation learning [62]. The model architecture includes three convolutional layers for feature extraction, followed by five fully connected layers. DAVE-2 has been extensively used in a variety of ADAS testing studies [13], [63], [16], [64], [65], [66]. The model takes as input an image representing a road scene, and it is trained to predict vehicle's actuators commands. Our implementation includes a DNN with lane-keeping (LK) and adaptive cruise control (ACC) capabilities, as DAVE-2 is trained to conduct the vehicle on the right lane of the road at the maximum possible speed, by predicting appropriate steering and throttle commands.

### C. Experimental Platforms and Benchmarks

*1) Semantic Segmentation:* We test SegFormer using the Virtual KITTI dataset (vKITTI) [67], commonly used for autonomous driving research. It provides 21,260 photo-realistic frames across five of the 20 KITTI real-world scenarios (i.e. 01, 02, 06, 18, and 20) rendered using the Unity engine. It includes pixel-level segmentation ground truths (GT) and semantic labels for urban objects such as roads, traffic lights and vehicles. Each frame is available in six weather conditions: sunny (nominal), fog, morning, overcast, rain, and sunset. We divided the scenarios in vKITTI as follows:

- *Training set*: Scenarios 01, 02, and 06 (nominal weather), randomly split into $90\%$ and $10\%$ of the samples for training and validation of the SegFormer model for $\mathbf{RQ_1}$;
- *Augmentation set*: Scenario 18 (nominal weather), split similarly, used for fine-tuning SegFormer with perturbations for $\mathbf{RQ_2}$;
- *Testing set (N)*: Scenario 20 (nominal weather), used to evaluate perturbation disruptions for $\mathbf{RQ_1}$ and as a baseline for SegFormer variants in $\mathbf{RQ_2}$;
- *Testing set (W)*: Scenario 20 (with weather effects), used to evaluate SegFormer variants for $\mathbf{RQ_2}$.

*2) LK/ACC:* To evaluate the generalizability of our results across different simulation environments, we conducted experiments on both the Udacity [50] and the Sdsandbox Donkey Car™ [51] simulators.

Udacity [50] is developed with Unity 3D [68], a popular cross-platform game engine, based on the Nvidia PhysX engine [69], featuring discrete and continuous collision detection, ray-casting, and rigid-body dynamics simulation. Udacity also supports testing under various weather conditions, such as day/night, rain, snow, and fog, which we refer to as *naturalistic* perturbations because they simulate real-world phenomena like virtual artefacts (e.g., raindrops) and lighting variations (e.g., day/night transitions). In addition to these effects, in this study, we introduce four new weather conditions that are not based on particle effects but instead focus on lighting changes by altering the skybox and scene illumination intensity. These conditions, listed in order of increasing darkness, are named: dawn, moonshine, starry, and dark/overcast. Donkey Car™ includes a high fidelity digital twin of the Donkey Car, a 1:16 scale radio-controlled car with self-driving capabilities, used for ADAS testing research in physical environments [13], [70], [71]. We selected these platforms because they are open-source and suitable for Level 2 ADAS evaluation. However, this choice is not exclusive, and other simulators can be integrated in PerturbationDrive with additional engineering cost.

For both simulators, we use the following scenarios:

- *Training roads:* These roads are used to train the DAVE-2 model evaluated in both $\mathbf{RQ_1}$ and $\mathbf{RQ_2}$. The road structures are randomly generated using PerturbationDrive, incorporating a variety of curves, lengths, and curvatures to ensure diversity in the training set.
- *Testing roads$_{RQ1}$:* These manually designed scenarios consist of 10 road tracks with increasing difficulty, ranging from simple, straight roads to more complex, curvy paths, used to evaluate the model's performance in $\mathbf{RQ_1}$.

As we evaluate generalizability ($\mathbf{RQ_2}$) using only Udacity, since the Donkey Car simulator lacks weather variations, we include additional independent test sets:

- *Fine-tuning roads:* These roads are used to fine-tune the DAVE-2 model evaluated in $\mathbf{RQ_2}$, as such they are designed to differ from roads used to test the model's performance. We use PerturbationDrive to generate random roads with specific curvature ranges and road lengths which differ from the ones found in other road sets.
- *Testing roads$_{RQ2}$:* These roads are used to evaluate both the nominal and fine-tuned DAVE-2 models in $\mathbf{RQ_2}$. These scenarios include 15 road shapes, with different degrees of complexity, and the ability to set simulation-based weather effects. In particular, five of these scenarios drastically differ from the ones found in *Training roads*.

### D. Procedure

As one of the goals of our study is to evaluate the image perturbations for system level testing, we performed a preliminary analysis to assess their computational overhead. Excessive processing times could in fact jeopardize the execution of simulation-based tests and lead to spurious failures that are not related to the actual robustness of the ADAS. We evaluated the execution time of 250 iterations for each type of perturbation across multiple intensity levels, using random RGB images with a resolution of $240\times320$ pixels—consistent with the image dimensions used by the simulators under evaluation. This experiment was conducted using the `Pyperf` library [72] on a machine equipped with an Apple M1 processor. Since each perturbation must be applied to every simulation frame, we established an upper time limit for acceptable computation based on the simulators' frame rates.

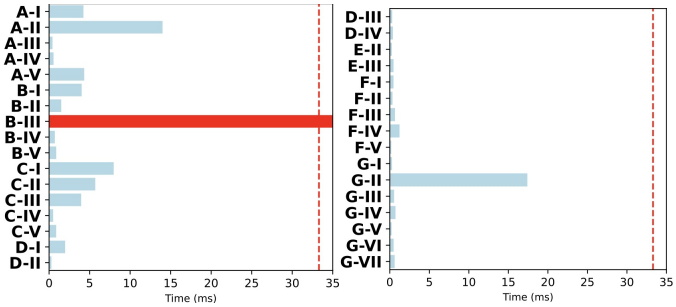



Fig. 2: Benchmarking perturbations.

The two simulators in our study, Udacity and Sdsandbox Donkey Car, operate at frame rates of 20 and 30 frames per second (fps), respectively, which correspond to frame intervals of 50 ms and 33.3 ms. To ensure consistency in comparisons, we adopted the higher frame rate of 30 fps, setting 33.3 ms as the maximum acceptable computation time per frame for all perturbations. This upper limit assumes that the time required for the ADAS system to process each frame is negligible.

Figure 2 reports the average execution time for each perturbation. Our study shows that the majority of perturbations are feasible for real-time evaluation, with most taking less than 10 ms to execute. Only Zoom blur (B-III) significantly exceeds the 33.3 ms threshold at 95.6 ms. As a result, Zoom blur is excluded from further evaluations.

*1) $RQ_1$:* **Semantic Segmentation.** We first fine-tune a pre-trained SegFormer model [73] for 10 epochs using the *training set* split of the vKITTI dataset (see Section III-C1) and the Adam optimizer with a learning rate of $6e^{-5}$.

Regarding the evaluation phase, we instructed PerturbationDrive to introduce controlled perturbations in the images of the *Testing set (N)*. Each image was perturbed across five different intensity levels. This approach enables us to systematically evaluate not only how different perturbation types affect the model's output but also at which intensity levels the impact becomes significant. We then execute the trained SegFormer model on both the nominal (unperturbed) and perturbed images of *Testing set (N)*. For each test image, the model generates a segmentation map, with each pixel classified into a corresponding semantic category. These predicted segmentation maps are then compared with the ground truth annotations from the dataset to assess the effect of the perturbations on the model's segmentation performance.

To quantify the effects of the perturbations, we calculate the Intersection over Union (IoU), chosen for its widespread use in evaluating segmentation model performance, particularly for driving scenes [57], [74]. IoU measures, for each semantic class, the overlap between the predicted and ground truth segmentation maps, calculated as the ratio of the intersection (where predicted and true segments match) to the union (the total area covered by both). This makes IoU especially useful for understanding how accurately the model identifies critical elements in driving environments, such as vehicles.

**LK/ACC.** We trained the DAVE-2 model in both simulators using data collected by a human driver on *Training roads*. We then validate the model using the *Testing roads*$_{RQ1}$. The behaviour of the ADAS under nominal conditions constitutes a baseline for evaluating the effects of perturbations.

Next, we use PerturbationDrive to inject perturbations into the images during simulation-based testing, on the same set of *Testing roads*$_{RQ1}$. The DAVE-2 model's performance on each perturbed road is then compared against the performance under nominal conditions. To quantify the performance degradation caused by perturbations, we evaluate several metrics. First, we quantified the *success rate* and *completion rate*. The first metric indicates the percentage of scenarios in which the LK-ACC system successfully reaches the end goal, while the second measures the extent of the scenario executed before either a failure or goal completion occurs. Next, we classify failures into two types: *Out of road (OR)*, when the vehicle leaves the designated driving lane, and *Out of time (OT)* triggered when misbehaviour causes a delay, leading to a 200-second timeout before the scenario is completed. For successful scenarios, we further analyze the *execution time*, which is useful for identifying the quality of throttle predictions and the *driving jitter*, which helps assess the steering quality by calculating the first derivative of the distance from the center of the lane (Cross-Track Error) and normalizing it by the lane width to indicate deviations in lane centering.

*2) $RQ_2$:* **Semantic Segmentation Dataset Augmentation.** We introduce controlled perturbations to the images from the dataset *Augmentation set* (Section III-C1), utilizing the perturbation types identified in $\mathbf{RQ_1}$ at maximum intensity.

After generating the perturbed images, we use them to perform fine-tuning of the SegFormer model, which was trained on the original vKITTI dataset in nominal conditions, by executing one epoch of training. To evaluate the impact of this augmentation, we test the fine-tuned model on dataset *Testing set* under both nominal conditions (*Testing set (N)*) and simulated weather effects (*Testing set (W)*). The goal is to measure whether the fine-tuned model exhibits improved generalization and robustness compared to the original, unmodified model. The same evaluation metric of $\mathbf{RQ_1}$ (i.e., IoU) will be used to compare the model's performance on both nominal and perturbed data.

**LK/ACC Online Continuous-learning.** We employ a hybrid control system consisting of a pure-pursuit controller for steering and a PID controller for throttle. The pure-pursuit controller calculates the steering angle to keep the vehicle aligned with the road by following predefined waypoints, while the PID controller regulates the throttle based on an expected speed and the car's relative distance to the center of the target lane. If the car deviates far from the center of the lane, the PID controller reduces the throttle to slow the vehicle down, allowing it to regain control and move back towards the center. This helps to prevent failures in challenging situations. If the car is close to the center of the lane, the throttle is adjusted to match the expected speed parameter, to obtain a vehicle speed consistent with the target speed. We first deploy

TABLE II: $RQ_1$: Effectiveness results for different types of perturbations for robustness testing of ADAS.

| | Semantic Segmentation | | | | LK/ACC | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | vKITTI | | | | Udacity | | | | | | | Donkey Car | | | | | |
| | | | | | Overall | | | | | Non-failing | | Overall | | | | Non-failing | |
| | IoU | | | | Success Rate | | | Fail Type | | Time | Jitter | Success Rate | | | Fail Type | | Time | Jitter |
| Perturbation | Avg | Std | Max | Min | Avg | Std | Trend | OR | OT | Avg | Avg | Avg | Std | Trend | OR | OT | Avg | Avg |
| nominal | 0.66 | - | - | - | 100% | - | - | 0 | 0 | 96.84 | 2.83% | 100% | - | - | 0 | 0 | 102.17 | 2.50% |
| **Noise** | | | | | | | | | | | | | | | | | | |
| A-I Gaussian noise | 0.43 | 0.16 | 0.64 | 0.23 | 86% | 3.00 | | 2 | 5 | 95.6 | 44.8% | 98% | 2.00 | | 0 | 1 | 83.3 | 9.4% |
| A-II Poisson noise | 0.50 | 0.10 | 0.58 | 0.32 | 100% | 0.00 | | 0 | 0 | 50.2 | 9.4% | 98% | 2.00 | | 1 | 0 | 52.2 | 9.0% |
| A-III Impulse noise | 0.41 | 0.16 | 0.62 | 0.21 | 96% | 3.00 | | 0 | 2 | 129.9 | 11.0% | 100% | 0.00 | | 0 | 0 | 95.5 | 7.6% |
| A-IV JPEG artifacts | 0.59 | 0.02 | 0.62 | 0.56 | 66% | 88.00 | | 1 | 16 | 149.2 | 38.8% | 98% | 2.00 | | 1 | 0 | 77.3 | 4.8% |
| A-V Speckle noise | 0.58 | 0.09 | 0.66 | 0.41 | 88% | 2.00 | | 6 | 0 | 59.2 | 20.0% | 100% | 0.00 | | 0 | 0 | 73.4 | 6.2% |
| **Blur and Focus** | | | | | | | | | | | | | | | | | | |
| B-I Defocus blur | 0.57 | 0.06 | 0.65 | 0.47 | 96% | 3.00 | | 1 | 1 | 103.4 | 12.6% | 100% | 0.00 | | 0 | 0 | 84.6 | 8.0% |
| B-II Motion blur | 0.63 | 0.03 | 0.66 | 0.58 | 84% | 8.00 | | 0 | 8 | 122.2 | 46.0% | 94% | 8.00 | | 3 | 0 | 93.8 | 8.6% |
| B-IV Gaussian blur | 0.58 | 0.08 | 0.66 | 0.46 | 94% | 3.00 | | 3 | 0 | 87.6 | 11.8% | 94% | 3.00 | | 3 | 0 | 99.9 | 9.0% |
| B-V Low-pass filter | 0.63 | 0.01 | 0.64 | 0.62 | 88% | 2.00 | | 6 | 0 | 88.0 | 28.6% | 96% | 3.00 | | 2 | 0 | 92.7 | 8.6% |
| **Weather** | | | | | | | | | | | | | | | | | | |
| C-I Frosted glass | 0.62 | 0.04 | 0.65 | 0.54 | 70% | 135.00 | | 15 | 0 | 52.1 | 11.6% | 94% | 3.00 | | 3 | 0 | 69.3 | 7.8% |
| C-II Snow | 0.64 | 0.04 | 0.67 | 0.56 | 30% | 120.00 | | 35 | 0 | 32.5 | 13.8% | 96% | 8.00 | | 2 | 0 | 74.9 | 10.2% |
| C-III Fog | 0.56 | 0.14 | 0.67 | 0.29 | 54% | 228.00 | | 23 | 0 | 52.2 | 10.6% | 90% | 5.00 | | 5 | 0 | 81.7 | 11.6% |
| C-IV Brightness | 0.66 | 0.00 | 0.66 | 0.65 | 92% | 7.00 | | 3 | 1 | 123.2 | 12.2% | 96% | 8.00 | | 2 | 0 | 95.0 | 10.4% |
| C-V Contrast | 0.66 | 0.00 | 0.66 | 0.65 | 90% | 20.00 | | 4 | 1 | 92.5 | 16.4% | 52% | 197.00 | | 17 | 7 | 91.8 | 13.8% |
| **Distortion** | | | | | | | | | | | | | | | | | | |
| D-I Elastic | 0.63 | 0.01 | 0.64 | 0.61 | 98% | 2.00 | | 0 | 1 | 73.3 | 9.6% | 100% | 0.00 | | 0 | 0 | 85.1 | 6.6% |
| D-II Pixellate | 0.62 | 0.05 | 0.66 | 0.53 | 96% | 8.00 | | 2 | 0 | 84.4 | 14.4% | 98% | 2.00 | | 1 | 0 | 96.6 | 8.6% |
| D-III Sample | 0.41 | 0.16 | 0.64 | 0.20 | 54% | 213.00 | | 21 | 2 | 86.6 | 29.6% | 90% | 15.00 | | 5 | 0 | 75.8 | 15.2% |
| D-IV Sharpen | 0.66 | 0.01 | 0.67 | 0.64 | 60% | 255.00 | | 20 | 0 | 64.0 | 18.6% | 94% | 8.00 | | 3 | 0 | 87.1 | 9.4% |
| **Affine Transformations** | | | | | | | | | | | | | | | | | | |
| E-II Scale | - | - | - | - | 70% | 45.00 | | 5 | 10 | 220.7 | 24.6% | 46% | 168.00 | | 15 | 12 | 123.7 | 15.0% |
| E-III Translate | - | - | - | - | 20% | 120.00 | | 29 | 11 | 323.8 | 22.2% | 64% | 58.00 | | 18 | 0 | 74.4 | 11.4% |
| **Graphic Patterns** | | | | | | | | | | | | | | | | | | |
| F-I Splatter | 0.65 | 0.01 | 0.66 | 0.64 | 58% | 122.00 | | 16 | 5 | 156.8 | 16.0% | 66% | 108.00 | | 14 | 3 | 90.3 | 11.8% |
| F-II Dotted lines | 0.66 | 0.01 | 0.66 | 0.65 | 92% | 7.00 | | 3 | 1 | 92.6 | 18.6% | 80% | 65.00 | | 5 | 5 | 108.2 | 8.2% |
| F-III ZigZag | 0.66 | 0.00 | 0.66 | 0.65 | 100% | 0.00 | | 0 | 0 | 77.2 | 17.4% | 100% | 0.00 | | 0 | 0 | 91.7 | 5.6% |
| F-IV Canny edges | 0.63 | 0.00 | 0.64 | 0.63 | 88% | 2.00 | | 6 | 0 | 90.0 | 19.4% | 100% | 0.00 | | 0 | 0 | 85.3 | 7.2% |
| F-V Cutout | 0.64 | 0.02 | 0.66 | 0.60 | 84% | 43.00 | | 8 | 0 | 123.4 | 12.2% | 66% | 33.00 | | 15 | 2 | 99.8 | 14.6% |
| **Color/Tone Adjustments** | | | | | | | | | | | | | | | | | | |
| G-I False color | 0.50 | 0.09 | 0.58 | 0.34 | 24% | 153.00 | | 38 | 0 | 40.8 | 14.4% | 82% | 92.00 | | 5 | 4 | 89.5 | 12.4% |
| G-II Phase scrambling | 0.50 | 0.16 | 0.66 | 0.23 | 56% | 113.00 | | 19 | 3 | 44.3 | 33.4% | 46% | 213.00 | | 21 | 6 | 49.0 | 15.2% |
| G-III Histogram eq. | 0.65 | 0.01 | 0.66 | 0.64 | 52% | 32.00 | | 6 | 18 | 197.5 | 49.0% | 74% | 28.00 | | 13 | 0 | 75.0 | 9.6% |
| G-IV White balance | 0.66 | 0.00 | 0.66 | 0.66 | 94% | 8.00 | | 0 | 3 | 91.9 | 29.6% | 92% | 2.00 | | 4 | 0 | 92.2 | 8.0% |
| G-V Greyscale | 0.66 | 0.02 | 0.67 | 0.62 | 66% | 173.00 | | 13 | 4 | 143.8 | 32.0% | 88% | 17.00 | | 6 | 0 | 85.5 | 11.8% |
| G-VI Saturation inc. | 0.65 | 0.01 | 0.66 | 0.64 | 68% | 72.00 | | 11 | 5 | 118.8 | 22.4% | 46% | 258.00 | | 0 | 27 | 131.5 | 6.2% |
| G-VIb Saturation dec. | 0.64 | 0.03 | 0.66 | 0.60 | 52% | 187.00 | | 15 | 9 | 103.6 | 17.6% | 88% | 7.00 | | 6 | 0 | 91.7 | 5.8% |
| G-VII Posterize | 0.65 | 0.02 | 0.66 | 0.62 | 82% | 32.00 | | 4 | 5 | 108.1 | 40.2% | 86% | 28.00 | | 7 | 0 | 78.8 | 14.6% |

the pure-pursuit/PID combo on the *Training roads* set under nominal conditions. This enables us to gather accurate ground truth data that represents good, failure-free driving behaviour. Once the data collection is complete, we train the DAVE-2 model using this nominal dataset. Hereafter, this model will be referred to as *DAVE-2 (N)*.

To establish the baseline for how well the model performs without additional training on perturbed scenarios, we evaluate *DAVE-2 (N)*'s performance on a set of test roads that differ in topology from the training set (*Testing roads*$_{RQ2}$), both in nominal weather conditions and under real weather scenarios in the Udacity simulator. To observe both generalization and robustness improvements, the set of 15 roads (*Testing roads*$_{RQ2}$) has been designed so that *DAVE-2 (N)* succeeds in 10 out of 15 scenarios in nominal conditions. These tests form the basis for assessing the model's performance without exposure to perturbed environments. Next, we apply the image perturbations identified in **RQ$_1$** across randomly generated roads, using PerturbationDrive to create two new random roads (*Fine-tuning roads*) that introduce different driving challenges from those encountered during the nominal evaluation. These random roads are designed to test the model's robustness under varied and unforeseen conditions. During this phase, we apply five intensity levels for each type of image perturbation. While running *DAVE-2 (N)* to evaluate its robustness under perturbed environments, the pure-pursuit/PID expert driver operates in *shadow mode*, continuously collecting ground truth data.

With the additional perturbed data collected from the randomly generated roads, we conduct one epoch of fine-tuning on the DAVE-2 model using this new dataset, obtaining *DAVE-2 (FT)*. Finally, we re-evaluate the *DAVE-2 (FT)* model on both nominal roads and real-world weather conditions in the Udacity simulator. To measure the impact of continuous learning, we compare *DAVE-2 (N)* and *DAVE-2 (FT)* models' behaviour using the same metrics used to answer **RQ$_1$**.

### *E. Results*

**RQ$_1$ (effectiveness).** Table II shows our effectiveness results, for both ADAS. Concerning semantic segmentation, The left side of Table II presents the SegFormer model's performance on the *Testing set (N)* under nominal and perturbed conditions, reporting the average per-class IoU (i.e., the average of the

TABLE III: $RQ_2$: Average metrics for original and extended models on testing datasets.

| Semantic Segmentation | | | LK/ACC | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Overall | | | | | | Non-failing | | | |
| | Average IoU | | | Success rate (%) | | # OR | | # OT | | Time (s) | | Jitter (%) | |
| Weather | Original | Extended | Weather | N | FT | N | FT | N | FT | N | FT | N | FT |
| nominal | 0.663 | 0.718 | nominal | 64 | 78 | 5 | 3 | 0 | 0 | 28.30 | 34.90 | 2.32 | 5.20 |
| fog | 0.365 | 0.520 | fog | 14 | 57 | 12 | 6 | 0 | 0 | 24.50 | 26.60 | 4.18 | 3.93 |
| morning | 0.667 | 0.709 | dawn | 57 | 64 | 6 | 5 | 0 | 0 | 73.20 | 24.65 | 2.56 | 4.47 |
| overcast | 0.669 | 0.707 | dark/overcast | 5 | 85 | 7 | 2 | 0 | 0 | 37.25 | 24.40 | 2.66 | 3.33 |
| rain | 0.385 | 0.650 | rain | 42 | 78 | 8 | 3 | 0 | 0 | 57.05 | 26.60 | 3.08 | 3.35 |
| sunset | 0.685 | 0.748 | moonshine | 21 | 78 | 10 | 3 | 1 | 0 | 98.35 | 24.65 | 2.68 | 3.98 |
| – | – | – | snow | 35 | 78 | 9 | 3 | 0 | 0 | 48.85 | 27.10 | 3.08 | 4.39 |
| – | – | – | starry | 21 | 78 | 10 | 3 | 1 | 0 | 98.15 | 26.90 | 2.78 | 3.29 |

IoU for each class) calculated at each of the five intensity levels. We use the average (Avg.), standard deviation (Std.), maximum (Max.), and minimum (Min.) statistics derived from these values to reflect overall performance and variability.

The Avg. IoU gives an overall assessment, while Std. shows variability in performance based on intensity. Max. and Min. IoU show the least and the most effective perturbation intensities respectively. Perturbations A-III and D-III were the most disruptive, with Avg. IoUs of $0.41$ (-38%), closely followed by A-I at $0.43$ (-35%). In contrast, perturbations like B-V, C-IV, C-V, and the Graphic patterns (F) category had little to no effect, as indicated by consistently low Std. and high Avg. values. At the highest intensities, D-III and A-III reduced IoU to $0.20$ (-70%) and $0.21$ (-68%), respectively, while A-I, G-II, and others produced minimum IoUs below $0.35$ (-47%). Perturbations with high Std. values, such as G-II, showed a wide variance in their impact, with disruption ranging from 0% to 65%, depending on intensity.

Concerning LK/ACC, the right side of Table II shows the DAVE-2 model effectiveness under various perturbations in the Udacity and Donkey Car simulators. We report the average success rate, standard deviation, completion rate trends over the five intensities as a histogram, and failure types—either Out of Road (OR) or Out of Time (OT)—across five intensity levels. For non-failing scenarios, execution time and driving jitter are provided to assess the impact on throttle and steering. Eight perturbations had minimal impact, reducing success rates by less than 10% in both simulators (e.g., A-II, A-III, B-I, B-IV). Five others reduced success rates by less than 20%. Nine perturbations affected simulators differently, reducing success rates by less than 20% in Donkey Car but by more than 20% in Udacity (e.g., A-IV, C-I, D-III). Perturbation E-II, G-II, and G-VI had the largest effect in Donkey Car, lowering success rates to 46%, while E-III, G-I, and C-II were most disruptive in Udacity, reducing success rates to as low as 20%.

Failure types were mostly OR, but some perturbations, like A-I and A-IV, caused more OT failures, especially in Udacity. Perturbations causing OT failures generally increased execution time, while those causing OR failures reduced it, sometimes significantly, as seen with C-I, C-II, and C-III in Udacity. Driving quality also varied, with higher driving jitter in Udacity, indicating a less stable model. For this metric, the most disruptive perturbations were D-III, E-II, and G-VII for D, and A-I, A-IV, and G-VIII for Udacity.

Finally, the driving jitter is significantly higher in Udacity, indicating less stable driving. In Donkey Car, the most disruptive perturbations are D-III, E-II, F-V, G-II, and G-VII, while in Udacity, they are A-I, A-IV, B-II, G-III, and G-VIII, with no clear overlap between domains.

> **$RQ_1$ (effectiveness):** For both ADAS tasks, most image perturbations impact the robustness, though the effects of the same perturbation type vary across different tasks and ADAS models. For the semantic segmentation task, the most significant impact (-70%) was observed with the sample (D-III) perturbation, while in the LK/ACC task the more robust model was most affected (-54%) by phase-scrambling (G-II).

**$RQ_2$ (generalization).** Table III shows the generalization results. Concerning semantic segmentation, the leftmost section of Table III details the effectiveness of the SegFormer model, trained on the *Training set*, either in its original form (original) or after fine-tuning with image perturbations from the *Augmentation set* (extended). The model's performance is tested on images from both the *Testing set (N)* (nominal conditions) and the *Testing set (W)* (weather domains). For each scenario, we report the average per-class Intersection over Union (IoU), allowing a direct comparison of the model's performance between the original and extended versions.

The results show that fine-tuning the SegFormer model with augmented data improves its effectiveness across all weather conditions, particularly in challenging environments like fog and rain, as the IoU increases from $0.36$ to $0.52$ (44%) and $0.38$ to $0.65$ (71%), respectively. Nominal conditions also see an improvement from $0.66$ to $0.78$ (18%), with moderate gains in morning, overcast, and sunset scenarios.

Concerning LK/ACC, the evaluation results in the right section of Table III show that the effectiveness of DAVE-2, initially trained on the *Training roads* (*DAVE-2 (N)*) and then fine-tuned on the *Fine-tuning roads* (*DAVE-2 (FT)*). Each row provides the evaluation of the 15 *Testing roads*$_{RQ2}$, both under nominal conditions (row 1) and simulator-based weather conditions. The table compares the model effectiveness using the success rate (with 100% representing success on all 15 roads) and reports the number of failures, categorized by type: Out of Road and Out of Time. For non-failing scenarios, we also provide the average execution time and driving jitter.

DAVE-2 (FT) shows a significant improvement in both success rates and reduction in OR failures compared to DAVE-2 (N) across all weather conditions. In nominal conditions, the success rate increases to $78\%$ from an initial $64\%$, while the number of failures decreases from $5$ to $3$, indicating a model that generalizes better to new roads.

In foggy conditions, the success rate increases from $14\%$ to $57\%$, with OR failures decreasing from $12$ to $6$. For dawn, the success rate improves from $57\%$ to $64\%$, and OR failures drop from $6$ to $5$. In dark/overcast conditions, the success rate increases from $5\%$ to $85\%$, and OR failures fall from $7$ to $2$. In rainy conditions, the success rate rises from $42\%$ to $78\%$, with OR failures dropping from $8$ to $3$. Similarly, in moonshine, the success rate increases from $21\%$ to $78\%$, while OR failures reduce from $10$ to $3$. In snow, the success rate improves from $35\%$ to $78\%$, and OR failures drop from $9$ to $3$. Finally, in starry conditions, the success rate climbs from $21\%$ to $78\%$, with OR failures decreasing from $10$ to $3$.

In terms of OT failures, only moonshine and starry weather caused one OT failure each, which have been both mitigated during fine-tuning. Driving jitter shows a slight increase in most weather conditions after fine-tuning. In dawn, jitter rises from 2.56% to 4.47%; in snow, it increases from 3.08% to 4.39%. The increase in jitter is generally minimal, with the exception of fog, where it shows a slight improvement, decreasing from 4.18% to 3.93%.

**RQ$_2$ (generalization):** For both ADAS tasks, fine-tuning the DNN using image perturbations, improves the ADAS effectiveness on unseen, simulated, weather domains, while retaining the original capabilities on nominal scenarios.

## IV. Discussion

### A. Effectiveness (RQ$_1$)

Our study shows that image perturbations significantly impact the performance of both modular (semantic segmentation) and end-to-end (LK/ACC) ADAS systems, with varying effectiveness based on perturbation type and intensity. In the offline evaluation of the SegFormer model using the vKITTI dataset, perturbations like D-III Sample and A-III Impulse noise notably degraded performance, reducing IoU from $0.66$ to below $0.21$ at higher intensities. This confirms the vulnerability of vision-based models to visual distortions, even with advanced architectures like transformers. In contrast, perturbations such as Histogram equalization (G-III), White balance (G-IV), and all Graphic patterns (F) had little to no impact, suggesting a greater robustness to global adjustments or artificial patterns.

For the end-to-end LK/ACC system, the perturbations caused more failures in the Udacity simulator compared to the Donkey Car simulator, likely due to the distinct car dynamics between the simulators as Udacity utilizes wheel friction to move the car, whereas Donkey Car employs a kinematic model that directly translates the car's position based on inputs and current movement. Perturbation False color (G-I), for example, resulted in an average success rate of only 24% in the former, while 82% in the latter. This highlights the importance of employing different testing environments and simulation platforms for cross-validating research results in ADAS testing, as DNN models may behave differently between simulators [63], [75], [76]. An interesting finding of our study is that perturbations impacted the outputs of the ADAS differently. Most OR failures were triggered by steering errors of the LK system resulting from visual distortions, while perturbations like Scale (E-II) and Saturation (G-VI) affected the ACC system and led to OT failures.

### B. Generalization (RQ$_2$)

Fine-tuning the semantic segmentation model with perturbation-augmented data significantly improved performance across all weather conditions, particularly in fog and rain, where IoU increased from $0.36$ to $0.52$ and from $0.38$ to $0.65$. This shows that even common, arguably non-realistic, perturbations enhance resilience to more naturalistic real-world environmental changes. The robustness in nominal conditions also improved, asserting that our retraining pipeline increased the *overall* robustness of the ADAS rather than overfitting it to the new conditions.

Similarly, fine-tuning LK/ACC through continuous learning with real-time perturbations increased the success rates in all weather scenarios, with an increase of up to $80\%$ in the most challenging condition (i.e., dark/overcast skybox), with fewer OT failures. However, our findings also indicated an increase in driving jitter, resulting in a decrease in control smoothness for the ADAS post-retraining. These results suggest that future work for system-level robustness testing of ADAS should be directed toward balancing both functional and non-functional requirements, as achieving enhanced robustness in novel conditions should not come at the cost of reduced steering precision and a less smooth driving experience.

### C. Threats to Validity

*1) Internal validity:* Several factors may affect the internal validity of our study, particularly in the design and execution of the experiments. The use of two simulators with differing car dynamics and distinct LK/ACC models could introduce variations in performance unrelated to image perturbations. To account for this, we report nominal model performance in each evaluation step.

Our computational benchmarks for real-time feasibility were conducted on an Apple M1 processor. While consumer-grade hardware was chosen to reflect practical applications, differences in hardware specifications could affect execution times and perturbation feasibility in other setups.

We used IoU for semantic segmentation and success rates for LK/ACC as primary metrics, aggregated across perturbation intensity levels and scenarios. This aggregation might overlook specific effects at different intensities, which is why full experimental logs are provided in the replication package for more detailed analysis.

*2) External Validity:* Our system-level experiments were limited to two simulators and focused on Level 2 ADAS, which may limit the applicability of our findings to other simulators and higher levels of autonomy (e.g., Level 3 and Level 4 ADAS in CARLA [77]). However, studies have shown that there are dozens of simulation platforms available, both commercially and open-source [78], [79], with no consolidated omni-comprehensive solution. While our study shows that the magnitude and occurrence of failures do change across simulators, this choice does not undermine the core insights of our study. We will direct further research involving modular [80] or multi-modal-language-based ADAS [81] could offer valuable perspectives on the robustness, or potential vulnerabilities, of future-oriented ADAS technologies.

Our evaluations were conducted in simulated environments, which may not fully replicate the complexities of real-world driving conditions. While our perturbations mimic plausible visual distortions (e.g., weather conditions, noise, and lighting changes) and the simulated weather domains do represent realistic phenomena, real-world driving environments involve more complex scenarios and real-world sensors are subject to hardware-specific noise and physical degradation that cannot be fully simulated. As a result, the robustness gains observed in $RQ_2$ using simulation may not directly translate to improvements in real-world environments.

*3) Reproducibility:* The entire pipeline used to obtain the results discussed in this work, including model training, our library PerturbationDrive, metric calculations, and results, is available and can be reproduced [39].

## V. Related work

### A. Model-level Studies

Most research effort has been directed towards robustness testing of image classifiers, proposing benchmark datasets of corrupted images, such as ImageNet-C [7], MNIST-C [43], and FMNIST-C [82] or augmentation techniques to enhance the diversity of training data, such as RandAugment [45] and AugMix [8]. In contrast, we focus on ADAS, where a lack of DNN robustness can lead to safety-critical failures.

In the ADAS domain, model-level testing efforts predominantly target a restricted range of perturbations. Tools such as DeepXplore [15] use neuron coverage metrics to uncover misbehaviours in DNNs by applying only a few image perturbations types, such as lighting effects and occlusion by single or multiple small rectangles. DeepTest [16] employs perturbation types like rotation, translation, and shear, which are unlikely to induce realistic ADAS misbehaviours since they do not correspond to realistic driving scenarios.

Differently, in our work, we retrieved the most complete list of image perturbations from existing literature. We filtered out those that do not produce valid driving images, as well as those that are too computationally expensive for system-level testing. Furthermore, our research extends beyond robustness evaluation by investigating the potential of common image perturbations to enhance the generalizability of ADAS during domain adaptation and retraining campaigns.

### B. System-level Studies

Among the adversarial attack techniques, Wu et al. [35] developed a real-time adversarial attack on an end-to-end driving model, which can force the vehicle to deviate from its designated lane. Similarly, Yoon et al. [36] introduced an online image attack framework, which utilizes a binary decision boundary to decide when to launch attacks.

Other research leveraged image perturbations to create efficient runtime performance prediction modules for ADAS, applying these perturbations at a system level to evaluate potential errors rather than to execute driving maneuvers. For example, Luan et al. [38] introduce nine specific perturbations to each input frame, and calculates an anomaly score by comparing the driving commands from perturbed and unperturbed images. MarMot [14] implemented a runtime monitoring framework using five domain-specific metamorphic relations to influence the ADAS output, allowing for the generation of confidence scores by comparing the predictions from original and altered images at each frame. DeepManeuver [37] proposes a state-aware robustness testing framework, using road perturbations to expose failures in end-to-end driving models.

Our study complements these efforts by focusing on both modular and end-to-end ADAS, applying common perturbations during system-level evaluation using two simulators. While existing works only target adversarial attacks, specific perturbation types, and prediction reliability, our research systematically applies a broad range of image perturbations, road tracks, and multiple metrics, including success rates, failure types, and driving jitter, and domain generalization.

## VI. Conclusions and Future Work

Our study systematically evaluates the robustness of vision-based ADAS for perception tasks using 32 image perturbations from existing literature and assessing their impact at both the model level (semantic segmentation) and system level (LK/ACC). We found that perturbations such as Phase scrambling were particularly disruptive, significantly reducing performance across domains, while others such as ZigZag patterns had minimal effects. This evaluation highlights that perturbations can affect ADAS differently depending on whether they target offline perception tasks or real-time driving scenarios. Our findings further demonstrate that applying perturbation-based data augmentation and continuous learning improves ADAS robustness, particularly in adverse

weather conditions, such as fog and rain. This approach not only increased robustness to real-world environmental changes but also improved generalization on new scenarios under nominal conditions, emphasizing the value of perturbations in enhancing perception models' robustness.

In our future work we will investigate white-box adaptive perturbations that are informed by the confidence of the ADAS, or its attention patterns. We will also consider extending the study to additional ADAS and autonomous driving stacks, such as Autoware [83] or Apollo [84]. Moreover, we will evaluate PerturbationDrive for inducing perturbations beyond simulation environments, considering physical vehicles.

## Acknowledgements

This research was funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy.

## References

[1] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y. Li, L. Ma, Y. Xue, and Y. Liu, "A survey on automated driving system testing: Landscapes and trends," *CoRR*, vol. abs/2206.05961, 2022.

[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. 58 443–58 469, 2020.

[3] Y. Li and L. Xu, "Panoptic perception for autonomous driving: A survey," 2024.

[4] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[5] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," 2016.

[6] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, "Generalisation in humans and deep neural networks," 2020.

[7] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," 2019.

[8] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," 2020.

[9] E. Rusak, L. Schott, R. S. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel, "A simple way to make neural networks robust against diverse image corruptions," 2020.

[10] J. Laermann, W. Samek, and N. Strodthoff, *Achieving Generalizable Robustness of Deep Neural Networks by Stability Training*. Springer International Publishing, 2019, p. 360–373.

[11] E. Rusak, L. Schott, R. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel, "Increasing the robustness of dnns against image corruptions by playing the game of noise," 01 2020.

[12] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 IIPhDW*, 2018, pp. 117–122.

[13] A. Stocco, B. Pulfer, and P. Tonella, "Mind the Gap! A Study on the Transferability of Virtual Versus Physical-World Testing of Autonomous Driving Systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 04, pp. 1928–1940, apr 2023.

[14] J. Ayerdi, A. Iriarte, P. Valle, I. Roman, M. Illarramendi, and A. Arrieta, "Metamorphic runtime monitoring of autonomous driving systems," 2023.

[15] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–18.

[16] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 303–314.

[17] H. Zhou, W. Li, Y. Zhu, Y. Zhang, B. Yu, L. Zhang, and C. Liu, "Deepbillboard: Systematic physical-world testing of autonomous driving systems," 2018.

[18] F. U. Haq, D. Shin, S. Nejati, and L. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *Proceedings of 13th IEEE International Conference on Software Testing, Verification and Validation*, ser. ICST '20. IEEE, 2020.

[19] F. Ul Haq, D. Shin, S. Nejati, and L. Briand, *Empirical Software Engineering*, 2021.

[20] A. Stocco, B. Pulfer, and P. Tonella, "Model vs system level testing of autonomous driving systems: a replication and extension study," *Empirical Software Engineering*, vol. 28, no. 3, p. 73, May 2023.

[21] N. Neelofar and A. Aleti, "Identifying and explaining safety-critical scenarios for autonomous vehicles via key features," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 4, Apr. 2024.

[22] ——, "Towards reliable ai: Adequacy metrics for ensuring the quality of system-level testing of autonomous vehicles," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024.

[23] V. Crespo-Rodriguez, Neelofar, and A. Aleti, "Pafot: A position-based approach for finding optimal tests of autonomous vehicles," in *Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test (AST 2024)*, ser. AST '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 159–170.

[24] C. Lu, S. Ali, and T. Yue, "Epitester: Testing autonomous vehicles with epigenetic algorithm and attention mechanism," *IEEE Transactions on Software Engineering*, pp. 1–19, 2024.

[25] C. Lu, T. Yue, M. Zhang, and S. Ali, "Deepqtest: Testing autonomous driving systems with reinforcement learning and real-world weather data," 2023.

[26] Q. Pan, T. Wang, P. Arcaini, T. Yue, and S. Ali, "Safety assessment of vehicle characteristics variations in autonomous driving systems," 2023. [Online]. Available: https://arxiv.org/abs/2311.14461

[27] F. Klück, Y. Li, J. Tao, and F. Wotawa, "An empirical comparison of combinatorial testing and search-based testing in the context of automated and autonomous driving systems," *Information and Software Technology*, vol. 160, p. 107225, 2023.

[28] F. Klück, D. Sumann, and F. Wotawa, "Utilizing genetic algorithms for generating critical scenarios for testing autonomous driving functions," in *2024 IEEE International Conference on Artificial Intelligence Testing (AITest)*, 2024, pp. 73–80.

[29] D. Humeniuk, F. Khomh, and G. Antoniol, "Ambiegen: A search-based framework for autonomous systems testingimage 1," *Science of Computer Programming*, vol. 230, p. 102990, 2023.

[30] J. Wu, C. Lu, A. Arrieta, T. Yue, and S. Ali, "Reality bites: Assessing the realism of driving scenarios with large language models," in *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering*, ser. FORGE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 40–51.

[31] P. Arcaini and A. Cetinkaya, "Crag – a combinatorial testing-based generator of road geometries for ads testing," *Science of Computer Programming*, vol. 238, p. 103171, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642324000947

[32] T. Laurent, S. Klikovits, P. Arcaini, F. Ishikawa, and A. Ventresque, "Parameter coverage for testing of autonomous driving systems under uncertainty," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, Apr. 2023. [Online]. Available: https://doi.org/10.1145/3550270

[33] F. Khan, H. Anwar, and D. Pfahl, "Simulation-based safety testing of automated driving systems," in *Product-Focused Software Process Improvement*, R. Kadgien, A. Jedlitschka, A. Janes, V. Lenarduzzi, and X. Li, Eds. Cham: Springer Nature Switzerland, 2024, pp. 133–138.

[34] ——, "A process for scenario prioritization and selection in simulation-based safety testing of automated driving systems," in *Product-Focused Software Process Improvement*, R. Kadgien, A. Jedlitschka, A. Janes, V. Lenarduzzi, and X. Li, Eds. Cham: Springer Nature Switzerland, 2024, pp. 89–99.

[35] D. Liu, J. Zhao, A. Xi, X. H. Chao Wang, K. Lai, and C. Liu, "Data augmentation technology driven by image style transfer in self-driving car based on end-to-end learning," *Computer Modeling in Engineering & Sciences*, vol. 122, no. 2, pp. 593–617, 2020.

[36] H.-J. Yoon, H. Jafarnejadsani, and P. Voulgaris, "Learning when to use adaptive adversarial image perturbations against autonomous vehicles,"

*IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4179–4186, 2023.
[37] M. von Stein, D. Shriver, and S. Elbaum, "Deepmaneuver: Adversarial test generation for trajectory manipulation of autonomous vehicles," *IEEE Transactions on Software Engineering*, vol. 49, no. 10, pp. 4496–4509, 2023.
[38] S. Luan, Z. Gu, and S. Wan, "Efficient performance prediction of end-to-end autonomous driving under continuous distribution shifts based on anomaly detection," *Journal of Signal Processing Systems*, vol. 95, no. 12, pp. 1455–1468, 12 2023.
[39] "Replication package," https://anonymous.4open.science/r/PerturbationDrive-BE31/README.md, 2024.
[40] "Scopus: Abstract and Citation Database," https://www.scopus.com, accessed: 2024-02-08.
[41] "arXiv: e-Print Archive," https://arxiv.org, accessed: 2024-02-08.
[42] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," 2020.
[43] N. Mu and J. Gilmer, "Mnist-c: A robustness benchmark for computer vision," 2019.
[44] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *2019 IEEE/CVF CVPR*, 2019, pp. 113–123.
[45] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," 2019.
[46] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *CoRR*, vol. abs/1811.12231, 2018.
[47] D. Bashkirova, B. Usman, and K. Saenko, "Adversarial self-defense for cycle-consistent gans," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.
[48] W. Zhang, "Generating adversarial examples in one shot with image-to-image translation gan," *IEEE Access*, vol. 7, pp. 151 103–151 119, 2019.
[49] 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 *Proceedings of the 33rd ACM/IEEE ASE*, ser. ASE '18. Association for Computing Machinery, 2018, p. 132–142.
[50] Udacity, "Udacity self-driving car simulator," https://github.com/udacity/self-driving-car-sim, 2021, accessed: [2024-01-15].
[51] T. Kramer, "Sdsandbox," https://github.com/tawnkramer/sdsandbox, 2022.
[52] U. D. of Transportation, "A framework for automated driving system testable cases and scenarios," https://rosap.ntl.bts.gov/view/dot/38824/dot_38824_DS1.pdf, 2018.
[53] ——, "Standing general order on crash reporting for level 2 advanced driver assistance systems," https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf, 2022.
[54] Baidu Inc., "Baidu apolloscapes dataset," https://apolloscape.auto/index.html, 2018, accessed: [2024-01-15].
[55] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Álvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *CoRR*, vol. abs/2105.15203, 2021.
[56] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on CVPR*, 2016.
[57] "Papers with code, cityscapes segmentation benchmarks." [Online]. Available: https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes
[58] S. C. Lambertenghi and A. Stocco, "Assessing quality metrics for neural reality gap input mitigation in autonomous driving testing," in *Proceedings of 17th IEEE International Conference on Software Testing, Verification and Validation*, ser. ICST '24. IEEE, 2024, p. 12 pages.
[59] J. Xiao, Z. Xu, S. Lan, Z. Yu, A. Yuille, and A. Anandkumar, "1st place solution of the robust vision challenge 2022 semantic segmentation track," 2022.
[60] G. Rizzoli, F. Barbato, and P. Zanuttigh, "Multimodal semantic segmentation in autonomous driving: A review of current approaches and future perspectives," *Technologies*, vol. 10, no. 4, 2022.
[61] A. Liu and Z. Wang, "Cv 3315 is all you need : Semantic segmentation competition," 2022.
[62] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." *CoRR*, vol. abs/1604.07316, 2016.
[63] M. Biagiola, A. Stocco, V. Riccio, and P. Tonella, "Two is better than one: Digital siblings to improve autonomous driving testing," 2023.
[64] G. Jahangirova, A. Stocco, and P. Tonella, "Quality metrics and oracles for autonomous vehicles testing," in *Proceedings of 14th IEEE International Conference on Software Testing, Verification and Validation*, ser. ICST '21. IEEE, 2021.
[65] 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 *Proceedings of ASE '18*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 132–142.
[66] M. Biagiola and P. Tonella, "Boundary state generation for testing and improvement of autonomous driving systems," 2023.
[67] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.
[68] "Unity," https://unity.com/, 2024, accessed: 11-01-2024.
[69] "Nvidia PhysX," https://developer.nvidia.com/physx-sdk, 2022.
[70] H. Zhou, X. Chen, G. Zhang, and W. Zhou, "Deep Reinforcement Learning for Autonomous Driving by Transferring Visual Features," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.
[71] A. Viitala, R. Boney, and J. Kannala, "Learning to Drive Small Scale Cars from Scratch," *CoRR*, vol. abs/2008.00715, 2020.
[72] Python Software Foundation, "pyperf," https://github.com/psf/pyperf, 2024, accessed: 2024-01-22.
[73] "nvidia/segformer-b0-finetuned-cityscapes-640-1280 · hugging face," Huggingface.co, 2017. [Online]. Available: https://huggingface.co/nvidia/segformer-b0-finetuned-cityscapes-640-1280
[74] Alhaija, Hassan, Mustikovela, Siva, Mescheder, Lars, Geiger, Andreas, Rother, and Carsten, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *IJCV*, 2018.
[75] M. H. Amini, S. Naseri, and S. Nejati, "Evaluating the impact of flaky simulators on testing autonomous driving systems," *Empirical Softw. Engg.*, vol. 29, no. 2, feb 2024.
[76] M. Borg, R. B. Abdessalem, S. Nejati, F.-X. Jegeden, and D. Shin, "Digital twins are not monozygotic–cross-replicating adas testing in two industry-grade automotive simulators," in *ICST '21*. IEEE, 2021.
[77] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: an open urban driving simulator," *CoRR*, vol. abs/1711.03938, 2017.
[78] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang, and M. Yang, "Choose your simulator wisely: A review on open-source simulators for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 5, p. 4861–4876, May 2024.
[79] Y. Koroglu and F. Wotawa, "Towards a review on simulated adas/ad testing," in *2023 IEEE/ACM International Conference on Automation of Software Test (AST)*, 2023, pp. 112–122.
[80] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.
[81] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K.-Y. K. Wong, Z. Li, and H. Zhao, "Drivegpt4: Interpretable end-to-end autonomous driving via large language model," 2024.
[82] M. Weiss and P. Tonella, "Simple techniques work surprisingly well for neural network test prioritization and active learning," in *Proceedings of the 31th ACM SIGSOFT*, 2022.
[83] "Autoware," https://autoware.org, 2024.
[84] "Baidu Apollo," https://github.com/ApolloAuto/apollo/, 2024.