ThirdEye: Attention Maps for Safe Autonomous Driving Systems

Andrea Stocco
Università della Svizzera italiana
Lugano, Switzerland
andrea.stocco@usi.ch

Paulo J. Nunes
Federal University of Pernambuco
Recife, Brazil
paulojnbp@gmail.com

Marcelo d'Amorim
Federal University of Pernambuco
Recife, Brazil
damorim@cin.ufpe.br

Paolo Tonella
Università della Svizzera italiana
Lugano, Switzerland
paolo.tonella@usi.ch

ABSTRACT

Automated online recognition of unexpected conditions is an indispensable component of autonomous vehicles to ensure safety even in unknown and uncertain situations. In this paper we propose a runtime monitoring technique rooted in the attention maps computed by explainable artificial intelligence techniques. Our approach, implemented in a tool called ThirdEye, turns attention maps into confidence scores that are used to discriminate safe from unsafe driving behaviours. The intuition is that uncommon attention maps are associated with unexpected runtime conditions.

In our empirical study, we evaluated the effectiveness of different configurations of ThirdEye at predicting simulation-based injected failures induced by both unknown conditions (adverse weather and lighting) and unsafe/uncertain conditions created with mutation testing. Results show that, overall, ThirdEye can predict 98% misbehaviours, up to three seconds in advance, outperforming a state-of-the-art failure predictor for autonomous vehicles.

ACM Reference Format:
Andrea Stocco, Paulo J. Nunes, Marcelo d'Amorim, and Paolo Tonella. 2022. ThirdEye: Attention Maps for Safe Autonomous Driving Systems. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3551349.3556968

1 INTRODUCTION

Autonomous driving systems (ADS) consist of an integration of established systems of adaptive cruise control, parking assistance, and autopilots into a unified functional unit [75]. Modern ADS are developed with increasing capabilities to act autonomously with little to no human input, using a perception-plan-execution strategy [75]. The perception part is typically delegated to deep neural networks (DNNs) which are capable of learning driving actions from labeled input-output samples [22]. For ADS, typical inputs consist of driving images, whereas the outputs are driving commands predicted by the DNN, such as the angle that the car must steer at to drive safely. The input space of ADS (i.e., all possible driving images) is huge and hard to cover adequately, even with automated testing techniques [1, 4, 5, 21, 49]. Consequently, one of the main challenges associated with deploying trustworthy ADS on public roads consists in their need to operate safely even in partially unknown and uncertain environments, which can result in unpredictable and hazardous situations. On the other hand, increased acceptance of such driverless vehicles requires a high degree of robustness also in the presence of non-modelled phenomena, uncertainties, as well as errors or inaccuracies at the sensor level [53].

Existing works have proposed DNN supervisors to build a safety envelope over a DNN to assess its level of dependability in operation [24, 25, 28, 34, 60, 69, 73, 77]. Generic solutions consist of measuring the distance of a given data point from the distribution of the training dataset [34], or through input validation frameworks based on internal inconsistencies [73], or prediction snapshots [69]. For ADS, specific runtime monitoring solutions have been proposed to mitigate system-level failures. Frameworks such as SelfOracle [60], DeepRoad [77], or DeepGuard [28] monitor the ADS as a black box and examine its behaviour in response to changeable environmental conditions, essentially only by considering the input images processed by the system.

The main limitations of these approaches are twofold. First, black-box solutions can only handle data-driven failures induced by significant changes (e.g., corruptions) in the input image that makes it fall beyond the distribution of the inputs on which the ADS has been trained (out-of-distribution, or OOD) [16]. Thus, they suffer from the inability to capture failures caused by an inadequate training of the DNN model or by bugs at the model level [26]. Second, black-box solutions are prone to false positives/negatives as their functioning is extraneous to the internal state of the system, which can lead to a discrepancy between the system being monitored and the monitor. Indeed, if the ADS and the corresponding monitor have different generalization capabilities, this can cause false alarms to be reported, or, worse, safety-critical failures to be missed, as also noticed in the original papers [28, 60, 77].

This paper investigates the problem of building a white-box ADS failure predictor. Although there are many methods to investigate the internal functioning of a DNN [20, 34, 69], this paper focuses on the attention maps produced by explainable artificial intelligence techniques (XAI). Attention maps [31, 50, 51] are post-training approaches that highlight the input pixels that influence the output predictions the most. While primarily used for comprehension and debugging of DNNs, Tjoa et al. [64] have provided empirical

ASE '22, October 10–14, 2022, Rochester, MI, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3556968

evidence of the informative content of heatmaps. In this work we leverage attention maps for failure prediction to maintain the reliability of the ADS within a safety net.

Our technique, implemented in a tool called ThirdEye, consists of a self-attention monitor for ADS that turns attention maps into XAI-driven scores used as a white-box confidence estimator of the system. More specifically, ThirdEye performs online monitoring capturing visual snapshots during the execution of ADS and leverages the visual information extracted from the attention maps to automatically identify conditions in which the system is unconfident. We show that attention snapshots offer clues about the reliability of the ADS; ThirdEye synthesizes such snapshots into a confidence score using different summarization strategies (i.e., average, average derivative over time, reconstruction loss). Our technique works in an unsupervised fashion: failure prediction is performed by setting a threshold over the nominal XAI-confidence scores using probability distribution fitting. Anomalous driving conditions are detected when the confidence scores decrease within a detection window that precedes the failure.

We have evaluated the effectiveness of ThirdEye on the Udacity simulator for self-driving cars [66], using ADS available from the literature and a diverse set of failures induced by adverse operational scenes and mutation testing-simulated malfunctions. In our experiments on +70 simulations accounting for more than 350 failures, ThirdEye was able to safely anticipate up to 98% of them, up to 3 seconds in advance, a 30% increase with respect to SelfOracle [60], a state-of-the-art black-box strategy from the literature. The improvement is particularly evident for failures induced by mutation testing: on average, ThirdEye anticipated 85% more failures caused by mutated driving models. ThirdEye also achieves a better trade-off between prediction of misbehaviours and false alarms, with an F_3 improvement up to 49%.

Our paper makes the following contributions:

Technique. A self-attention monitoring technique for ADS failure prediction based on attention maps produced by XAI techniques. Our approach is implemented in the publicly available tool ThirdEye [65]. To the best of our knowledge, this is the first solution that uses XAI techniques to estimate the confidence of a DNN-based ADS and to anticipate system-level failures.

Evaluation. An empirical study showing that the XAI-based confidence scores used by ThirdEye are a promising white-box confidence metric for failure prediction, outperforming the black-box approach of SelfOracle [60].

Dataset. A dataset of more than 350 out-of-distribution and mutation-testing-induced ADS failures, based on the Udacity simulator for self-driving cars. This dataset can be used to evaluate the performance of failure prediction systems for ADS.

2 BACKGROUND

2.1 Lane-keeping ADS

ADS benefit from data gathered by sensors, cameras, and GPS to perceive the environment and predict the vehicle's controls (i.e., steer, brake, acceleration) through advanced DNNs.

This paper focuses on ADS that perform *behavioural cloning*, i.e., the vehicle learns the *lane keeping* functionality from humanly-labeled driving samples. The lane-keeping component is vital for the safe deployment of DNN-based ADS. The U.S. Department of Transportation, National Highway Traffic Safety Administration (NHTSA) reported that off road failures are second in frequency and first in cost (+15B USD) [68].

Models such as NVIDIA's DAVE-2 [11] learn how to drive by discovering latent patterns within a training set of images collected when the driver is an expert human pilot, and by predicting the corresponding driving commands imitating the driving behaviour of the human. In its most simplified form, a lane-keeping ADS such as DAVE-2 can be seen as a function $f : \mathbb{R}^d \rightarrow [-25°, +25°]$ where d is the dimension of the input image $\boldsymbol{x} \in \mathbb{R}^d$ (e.g., for a 140×320 image, $d = 44800$ pixels) and the output is a vector y of length 1, e.g., a real number representing a (predicted) steering angle in the range $\left[-25°, +25°\right]$, where $-25°$ indicates steering full left, $+25°$ indicates steering full right, and 0 means no steering applied.[1]

2.2 Failure Conditions for lane-keeping ADS

The ISO/PAS 21448 Safety of the Intended Function (SOTIF) standard [29] mandates risk mitigation strategy to be implemented within ADS to reduce risks and hazards associated with malfunctioning behaviour. At NHTSA Level 4 (High Automation), a system monitor checks for emerging functional insufficiencies with the aim to keep a high functional quality also in extreme conditions [25, 69, 77]. The ADS should disengage if the monitor regards the current conditions as unsafe, requiring the human driver to take control of the vehicle.

Among the root causes for ADS failures (e.g., off road driving) SOTIF recalls external unknown and internal uncertain conditions [29]. External unknown conditions consist of "abnormal" inputs representing rare, unexpected, and possibly unsupported environmental events, for which no prior knowledge was available during the training of the ADS (e.g., a specific road type, or weather/lighting condition). The DNNs used within ADS are not invariant to severe data distribution changes and this can cause system-level failures. Internal uncertain conditions correspond instead to misbehaviors of the perception component caused by the bugs inherent to the DNN model, introduced during its development. Instances of such bugs include inadequate training data and suboptimal choice of the model's architecture or of the training hyper-parameters [26].

2.3 Black-box Unsupervised Failure Prediction

Despite standards such as SOTIF [29], in practice, enumerating all possible hazardous conditions for an ADS in a written requirement specification is a challenging, if not infeasible, endeavour. As a consequence, research has considered failure prediction models that can be trained with no supervision (i.e., no knowledge of the anomalies), and, to make them more applicable, with no need to access information of the main system (black-box) [24, 25, 28, 60].

A black-box unsupervised failure predictor analyzes inputs and assigns a suspiciousness score to them, which should be low (below a threshold) if the inputs are supported, or high (above a threshold)

[1]these values reflect the steering capability of an ADS in a driving simulator [66].

otherwise. Notable examples are one-class SVM [52], clustering [17], self-organizing maps [35], and autoencoders (AEs) [14].

The variational autoencoder (VAE) is the most popular AE architecture [14] as it is able to efficiently learn the probability distribution of a large amount of complex data (such as images) using variational inference [2]. The VAE is trained to minimize the distance between the original data and its low-dimensional reconstruction with metrics such as the Mean Squared Error (MSE). A low MSE indicates that the input has characteristics similar to those of the training set, whereas a high MSE indicates potentially an OOD sample. As such, VAEs are used in anomaly detection tasks [25, 28, 60], as well as automated validity checkers for DNNs [16].

The main limitations of black-box approaches, including VAEs, consist in their zero knowledge of the system's internal behaviour, thus they are designed to react only for failures induced either by the corruption of inputs, or by a large degree of out-of-distributioness. Indeed, for unknown inputs, the ADS is likely to make a sequence of inaccurate predictions that may ultimately lead to a system failure, because of the prediction errors accumulated over time [23].

2.4 Deep Neural Networks Explanation

Explaining the predictions of DNNs has been largely studied using several interpretation methods [37, 40, 50, 51, 56, 63, 76]. The survey by Tjoa and Guan [63] distinguishes three main categories, namely *verbal methods*, *signal methods*, and *saliency methods*.

Verbal methods such as decision sets [37] or encoder-decoder frameworks [40] have been adopted in NLP problems since they produce lexical statements that humans can interpret naturally.

Signal methods target the stimulation of individual neurons or collections of neurons in a DNN to reconstruct an image similar to the input, based on the partial information stored in the neurons. However, feature maps produced with methods such as guided backpropagation [56], or deconvolutional networks [76], are known for producing sparse heatmaps.

Saliency methods explain DNN predictions by attributing a negative or positive value to each input feature according to how much it influenced the prediction. For instance, LIME [47] is a black-box technique that understands classification networks' decisions by assessing how the predictions change in response to local perturbations of the input data. Other methods use decomposition of signals propagated by their algorithms and selectively re-arrange them to provide interpretable information. For example, GradCam [54] uses gradient back-propagation up to the last convolutional layer to explain classifiers. Differently, LRP [7] uses relevance scores that are decomposed such that the sum of the scores in each layer of the DNN will be equal to the output. LRP has the drawbacks of generating noisy explanations as well as very similar outputs for samples pertaining to different classes [32].

In this paper we consider the attention maps produced by the SmoothGrad algorithm [55]. Unlike LRP, SmoothGrad makes gradient-based explanations sharper by adding noise and averaging over these artificially created noisy gradients. Like GradCAM, attention maps consider the gradient of the output prediction with respect to the input pixels [31]. Unlike GradCAM, SmoothGrad also works with regression DNNs such as those of ADS.

3 MOTIVATING EXAMPLE

Attention maps are images where relevant locations correspond to hot color intensities (e.g., red/yellow), whereas irrelevant locations correspond to cold color intensities (e.g., blue).

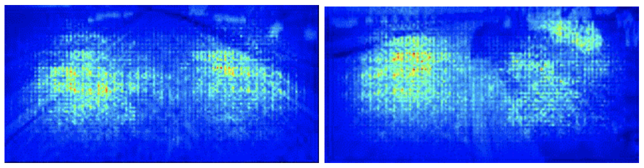

Figure 1: Attention map of an ADS during nominal driving (left), and a few seconds before an off-road failure (right).

Figure 1 shows an example in which attention maps—obtained with SmoothGrad [55]—are indicative of an upcoming failure of the ADS. In nominal conditions (left), the ADS focuses on foreground features that characterize the road. In this case, the attention map portrays two main clusters of attention, corresponding to the road's lanes. When driving in unsupported conditions (right), prior to a failure, the ADS is more uncertain. This is reflected in the attention map as only part of the attention still focuses on the road, while substantial attention is also paid to features in the background.

In the next section, we will describe our proposal for using attention maps by SmoothGrad to anticipate failures in conditions that cause the ADS to fail. Our technique aims to capture a *drop of DNN confidence by means of metrics derived from the attention maps*, considering single maps, consecutive maps, or map reconstruction based on nominal maps.

4 APPROACH

Our approach ThirdEye consists of two main phases, namely Training and Usage. In the first phase (Training, see Figure 2), ThirdEye automatically generates the attention maps for nominal driving instances of the ADS (see Section 4.1.1). Such attention maps are a visual snapshot of the ADS performance during nominal driving behaviour. ThirdEye is based on two intuitions: (1) attention maps derived during the processing of the inputs by the ADS are indicative of the confidence of the system [64], and (2) nominal and failure-inducing attention maps exhibit differences that can be captured by an anomaly detector.

ThirdEye turns the attention maps into XAI-driven confidence scores of the ADS using different summarization methods (Section 4.1.2). We consider a realistic setting, in which instances of failing driving behaviour cannot be sampled in any representative way, since failure conditions are potentially very diversified and partly unexpected. Hence, ThirdEye fits a probability distribution using only nominal scores. Then, it automatically estimates a threshold from such probability distribution (Section 4.1.3). This threshold is derived from the user defined permissiveness of the failure predictor to accept false alarms – i.e., from the tolerable false positive rate, a tunable parameter of our approach.

In the second phase (Usage, see Figure 4), ThirdEye is used along with the main ADS system to automatically predict whether the driving conditions are safe or unsafe, according to the attention maps retrieved during driving, and the threshold estimated during

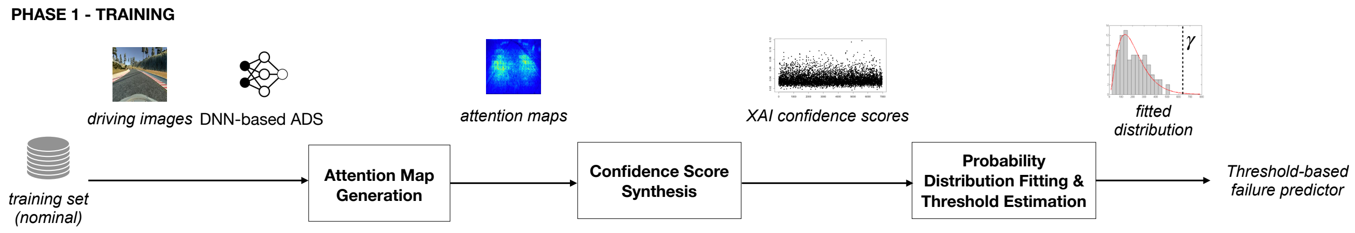

Figure 2: Training of ThirdEye.

the Training phase. If a driving condition is deemed as unsafe, ThirdEye warns the main driving component of the ADS (or the human driver). In the next sections, we describe each step of each phase in more detail.

4.1 Training of ThirdEye

4.1.1 Attention Map Generation. ThirdEye assumes having access to the training set $T = \{x_1, x_2, x_3, ..., x_n\}$ used to train the ADS, and to the trained ADS f. Our approach, however, does not need to modify the ADS model's architecture, nor to retrain it.

ThirdEye uses the XAI algorithm SmoothGrad [55] to retrieve an attention map for each driving image $x \in T$ used to train f. In particular, SmoothGrad produces an attention map $\boldsymbol{h} = \{h_p\}$ assigning each pixel p of x a value $\{h_p\} = \mathcal{H}(x, f, p)$ according to some function $\mathcal{H}$ derived from f. In SmoothGrad, the function $\mathcal{H}$ constructs $\boldsymbol{h}(x)$ by differentiating f with respect to the input x:

$$\boldsymbol{h}(x) = \partial f(x)/\partial x$$

The attention map $\boldsymbol{h}$ has the same dimensionality as $\boldsymbol{x}$ (i.e., width W, height H, and C channels) and represents how much difference a small change in each pixel of x would make to the prediction score of f. Since the derivative of the function f may fluctuate sharply at small scales [55], SmoothGrad uses a stochastic approximation by taking random samples in the neighbourhood of the input x, and averaging the resulting attention maps. Mathematically,

$$\hat{\boldsymbol{h}}(x) = \frac{1}{n} \sum \boldsymbol{h}(x + \mathcal{N}(\mu, \sigma^2))$$

To summarize, SmoothGrad (1) generates n versions of the image of interest by adding Gaussian noise to it, (2) it creates pixel attribution maps for all n versions of the image, and (3) it averages the pixel attribution maps. Averaging over multiple maps "smooths out" the derivative fluctuations. The output of the attention map generation step is a set $H = \{h_1, h_2, ..., h_n\}$ of attention maps for each image of the training set T.

4.1.2 Confidence Score Synthesis. ThirdEye uses three summarization functions to turn raw attention maps into XAI-driven confidence scores, namely average, derivative, and reconstruction loss.

Heatmap Average Function (HA). The first summarization function is based on the intuition that an attention map captured during nominal driving will have high relevance values (i.e., pixel intensities) focused on specific regions of interest (e.g., the lanes, see Figure 1), whereas attention will be more scattered and with lower pixel intensities during bad driving behaviour. As such, the HA function computes the average pixel intensity of an attention map.

Assuming attention maps have width W, height H and C channels (usually, RGB channels for colour images), notationally, for each attention map $\boldsymbol{h} \in H$, the average attention map score $\overline{\boldsymbol{h}}$ is computed as follows:

$$\overline{\boldsymbol{h}} = \frac{1}{WHC} \sum_{i=1,j=1,c=1}^{W,H,C} h_{[i][j][c]}$$

When applied to the whole training set T, the HA function returns the set of attention map average scores of each individual attention map in T, $HA = \{\overline{h_1}, \overline{h_2}, \overline{h_3}, ..., \overline{h_n}\}$.

Heatmap Derivative Function (HD). The second summarization function is based on the intuition that attention maps that do change frequently during driving could signal a poorly confident ADS. Thus, the HD function computes the average of the derivative of attention maps over time. Notationally, for two consecutive attention maps $\boldsymbol{h_{t-1}}, \boldsymbol{h_t} \in H$, the average of the derivative of attention map $\nabla \boldsymbol{h}$ is computed as follows:

$$\overline{\nabla \boldsymbol{h_t}} = \frac{1}{WHC} \sum_{i=1,j=1,c=1}^{W,H,C} h_{t[i][j][c]} - h_{t-1[i][j][c]}$$

The HD function returns the set of attention map average derivatives of each individual attention map in T (but the first), $HD = \{\overline{h_1}, \overline{\nabla h_2}, ..., \overline{\nabla h_n}\}$.

Heatmap Reconstruction Loss Function (HRL). The third summarization function uses reconstruction loss, i.e., it is based on learning a reconstruction function of latent features in the attention maps captured during nominal driving. A failure to reconstruct data during testing of the ADS can signal the presence of potentially failure inducing driving conditions.

The function HRL computes the reconstruction errors of the attention maps according to a variational autoencoder $\mathcal{V}$. The encoder of $\mathcal{V}$ encodes a given input $\boldsymbol{x} \in \mathbb{R}^d$ to a compressed representation $\boldsymbol{z} \in \mathbb{R}^z$ using a function $enc(\boldsymbol{x}) = \boldsymbol{z}$. The decoder of $\mathcal{V}$ decodes the encoded input with a reconstruction function $dec(\boldsymbol{z}) = \boldsymbol{x'}$, where $\boldsymbol{x'}$ is the reconstructed input $\boldsymbol{x}$. $\mathcal{V}$ minimizes a loss function $\mathcal{L}(\boldsymbol{x}, dec(enc(\boldsymbol{x})))$, which measures the distance between the original data and its low-dimensional reconstruction. Following existing guidelines [59], we set the dimension of the encoded representation z to 2 and used the mean squared error (MSE) as a loss function.

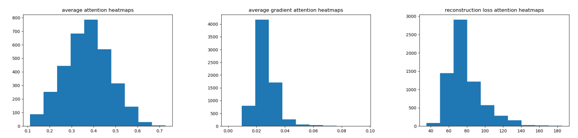

Figure 3: Examples of distributions of HA (left), HD (center), and HRL (right) XAI confidence scores.

Notationally, for each attention map $\boldsymbol{h} \in H$, the reconstruction error $\boldsymbol{h_e}$ is computed as follows:

$$\boldsymbol{h_e} = \mathcal{L}(\boldsymbol{h}, dec(enc(\boldsymbol{h})))$$

The HRL function returns the set of attention maps' reconstruction errors of each individual attention map in T, $HRL = \{h_{e1}, h_{e2}, h_{e3}, ..., h_{en}\}$.

Windowing of Confidence Scores. To mitigate the effect of individual single frame outliers, which are not expected to have a big impact on the performance of the ADS, ThirdEye applies a window function on non-overlapping, fixed length, sequences of scores. Two simple window functions are considered, one that computes the *maximum* score within a window, and a second that computes the *arithmetic mean* of the scores within a window. Window functions are applied to each of the proposed XAI confidence scores.

4.1.3 Probability Distribution Fitting & Threshold Estimation. The sets of (windowed) XAI confidence scores HA/HD/HRL represent a model of normality collected in nominal driving conditions using different synthesis methods from the attention maps.

To determine a threshold γ that sets the expected false alarm rate in nominal conditions below some configurable level, we use probability distribution fitting to obtain a statistical model of the XAI confidence scores.

In particular, γ is computed by (1) estimating the shape κ and scale θ parameters of a fitted Gamma distribution of the XAI confidence scores and (2) by selecting an acceptable false alarm rate [60]. We fit a Gamma distribution because the distributions of XAI scores contain strictly positive values (see Figure 3).

In this work, we set γ to the 95% percentile (i.e., we deem 5% as an acceptable false positive rate), in line with previous works [60]. For example, for HRL, $\gamma_{0.95} = p_{0.95}(\mathcal{L}(\boldsymbol{h}, g(f(\boldsymbol{h}))|\boldsymbol{h} \in W(H))$, where $W(H)$ represents the confidence scores after windowing.

4.2 Usage of ThirdEye

Figure 4 shows the second phase of our approach, Usage, in which ThirdEye is used as a runtime monitoring technique during the runtime execution of the ADS.

The ADS generates driving data that are processed by our approach. ThirdEye analyzes the incoming stream of driving images and attention maps are retrieved (Section 4.1.1). Next, confidence scores are synthesized from the attention maps (either HA/HD/HRL, see Section 4.1.2). When sufficient data samples are collected (e.g., matching the window size chosen during training, see Section 4.1.2), ThirdEye applies the window function to the stream (either max or mean). Each resulting score is compared against the threshold $\gamma_{0.95}$, which determines whether the windowed sequence of XAI confidence scores is to be regarded as anomalous. In such a case, a warning is sent to the ADS (or to the human driver); otherwise, ThirdEye keeps monitoring the next incoming driving frames.

4.3 Implementation

We implemented our approach in a Python tool called ThirdEye, which is available [65]. The tool supports ADS models written in Tensorflow/Keras, and it is integrated in the Udacity simulator for self-driving cars [66]. For computing the attention maps, ThirdEye leverages the SmoothGrad [55] implementation available in the toolkit `tf-keras-vis` [36].

5 EMPIRICAL EVALUATION

5.1 Research Questions

We consider the following research questions:
RQ$_1$ (effectiveness): How effective is ThirdEye at predicting failures of ADS? What is the best configuration?
RQ$_2$ (prediction over time): How does the prediction power of ThirdEye change when considering different detection periods?
RQ$_3$ (comparison): How does ThirdEye compare with SelfOracle [60], a failure predictor from the literature?

The first research question (RQ$_1$) aims to assess whether our approach is able to attain a high failure prediction rate, and which configuration (i.e., XAI confidence scores and window functions) yields the best prediction rate score. Ideally, failure prediction is only useful if it helps to anticipate a failure, which is studied in the the second research question (RQ$_2$). Lastly, to assess the effectiveness of our approach over existing solutions, the final research question (RQ$_3$) compares ThirdEye with the state-of-the-art failure predictor for ADS [60].

5.2 Testbed

We tested ThirdEye through simulation-based testing. The usage of simulation platforms is standard for testing ADS as simulator-generated data yield comparable conditions as the ones experienced in real world [23, 41, 57]. Moreover, driving simulators allow testing an ADS at the system level (online testing) because the DNN is embedded within the operational ecosystem in which the ADS is designed to operate. Testing the ADS only from the DNN model perspective (offline testing), disconnected from the ADS system, is not useful to expose the safety-critical failures that occur during in-field testing, such as the ones considered in this work (see Section 2.2).

As simulation platform, we used the Udacity simulator for self-driving cars [66], a cross-platform driving simulator developed with Unity3D [67], used in the ADS testing literature [28, 30, 48, 58–60]. The simulator supports various closed-loop tracks for testing behavioural cloning ADS models, including the ability to generate changeable driving scenarios (e.g., weather effects), which is useful to test an ADS on both nominal and unseen conditions.

In this paper, we chose the default sunny weather condition as the reference nominal scenario for our ADS models. Other choices of nominal condition are of course possible (e.g., snow). For unsupervised learning techniques such as ours (i.e., techniques that do

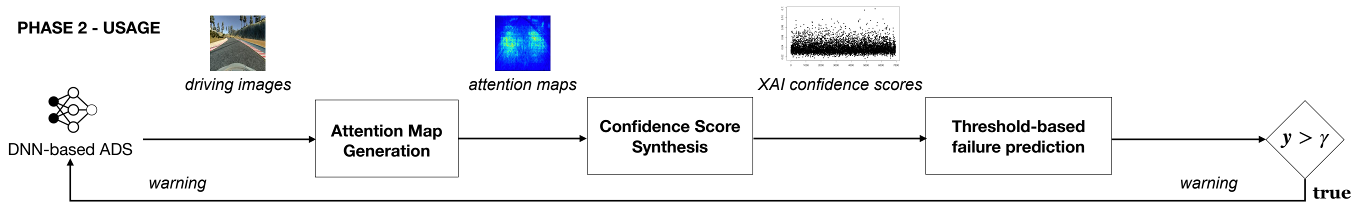

Figure 4: Usage of ThirdEye.

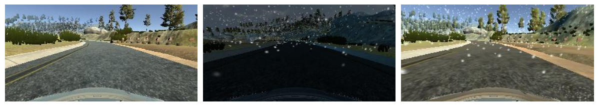

Figure 5: Examples of conditions from our evaluation set. Left: nominal (sunny). Center: OOD extreme (snow). Right: OOD moderate (snow).

not assume the availability of a representative set of anomalous conditions when training the failure predictor), the only requirement is that the chosen supported conditions are the same that are known at training time by the ADS.

5.3 Object of Study

To implement DNN-based ADS, we use Nvidia's DAVE-2 model [11], a reference model widely used as object of study in prior related work [27, 30, 46, 48, 60, 62, 77]. DAVE-2 consists of three 5x5 convolutional layers with stride 2 plus two 3x3 convolutional layers (no stride applied), followed by five fully-connected layers with dropout rate of 0.05 and ReLu activation function. We obtained the trained DAVE-2 model from the replication package of our baseline [60], to make sure to test the failure predictors using the same ADS used in previous work.

5.4 Procedure

5.4.1 Evaluation Set. We simulate the ADS testing practices customary of industry, where testers use a closed-loop track in a virtual environment, prior to on-road testing on public roads [6, 13, 70, 71]. We consider two kinds of scenarios for testing our failure predictor.

External unknown scenario. The first kind of testing scenario deals with failures induced by *out-of-distribution conditions* (OOD), exposing an ADS that has been trained on some given nominal conditions and environment to different instances of that environment. We use two OOD benchmarks in our study.

The first benchmark contains simulations provided by the replication package of the SelfOracle paper [60]. We refer to this benchmark as $\text{OOD}_{\text{extreme}}$ because it is characterized by severe illumination conditions w.r.t. the nominal sunny scenario (see Figure 5). It accounts for 21 simulations with different degrees of extreme OOD conditions: day/night, rain, snow, fog, day/night + rain, day/night + snow, day/night + fog. These conditions were created by (i) altering the environment's skybox (invisible ceiling object located at the boundary of the map) from sunny to adverse weather luminosity and by (ii) adding weather particles (snow or rain) rendered at runtime along the track.

We also consider a second benchmark of milder OOD conditions, called $\text{OOD}_{\text{moderate}}$, This second benchmark evaluates our approach considering weather effects only, i.e., without interferences due to adverse weather luminosity (see Figure 5). We deactivated the adverse weather luminosity skybox, while retaining a single unexpected weather condition at a time, namely *rain*, *fog*, or *snow*. We executed the DAVE-2 ADS varying the intensity of each condition in the range $\left[10\%, \ldots, 100\%\right]$, thus 10 times for each weather condition (day/night was discarded because non tunable).

Overall, concerning external unknown scenarios, a total of 51 OOD one-lap simulations were collected: 21 for $\text{OOD}_{\text{extreme}}$ and 30 for $\text{OOD}_{\text{moderate}}$ ($10 \times$ rain, $10 \times$ fog, $10 \times$ snow).

Internal uncertain scenario. The second kind of testing scenarios deals with *faulty ADS models* produced by automated mutation testing [27] that drive on the simulator under nominal conditions (sunny). Intuitively, these scenarios simulate the development process of an ADS model that has not been yet trained adequately. The third benchmark—referred to as Mutants—represent these scenarios. We obtained instances of mutated DAVE-2 models from the replication package of the DeepCrime mutation testing tool [27]. DeepCrime automatically mutates a DNN model using mutation operators designed to mimic real fault types occurring when developing DNNs, considering both data-level faults (e.g., wrongly labelled training data) and model-level faults (e.g., a suboptimal learning rate or dropout rate).

We executed all DAVE-2 mutants in the Udacity simulator and discarded those that were consistently failing (e.g., the corruption induced by a particular mutation operator caused severe malfunctions to the ADS driving from the very beginning of the simulation). A total of 20 one-lap simulations were retained that we confirmed to create internal uncertain scenarios (more details about the selected mutation operators are in our replication package [65].

Summary. Overall, our evaluation set comprises 349 failures that our approach is expected to predict and anticipate. Mutation testing caused most of the failures (66%), which is expected from a technique that systematically injects faults, whereas out-of-distribution conditions induced less failures (34%) as they were applied with different, increasing, levels of severity. Both scenarios are of interest for a failure predictor, which should be agnostic about the conditions that cause the failures (i.e., unknown inputs or DNN model bugs). To estimate the threshold used by ThirdEye, we finalized the evaluation data collection by performing three one-lap simulations under nominal sunny weather conditions (one for each of three benchmarks $\text{OOD}_{\text{extreme}}$, $\text{OOD}_{\text{moderate}}$, and Mutants) using the robust, unmutated, DAVE-2 model.

5.4.2 Detection Windows in Evaluation Set. The Udacity simulator automatically labels individual failing frames as either nominal or failing using a boolean flag, according to whether the ADS was on track or off-track, respectively. Since the goal of our framework is on predicting misbehaviors before they occur, we focus on the part of the simulation *preceding each failure*, whereas the frames labeled as failing are not considered.

Each simulation can exhibit multiple failures: in our evaluation strategy we assessed each failure individually. For each of them, we consider a detection window corresponding to one second of simulation in the Udacity simulator. We move the detection window from 1 to 3 seconds prior to the failures (time to failure, TTF for short). Studies on pre-crash automated seat belt systems [39, 81] indicate a range between 3 seconds to half a second as adequate TTF values for the activation of automated seat belt tightening. Also, according to previous studies in the Udacity simulator [59], a TTF of 3 seconds is deemed sufficient to avoid failures at 30 mph, which is the constant cruising speed of the ADS in the simulator.

5.4.3 ThirdEye's Configurations. We evaluate six configurations of ThirdEye. For SmoothGrad [55], we use the same hyper-parameters suggested in the original paper, specifically a noise level of 20% and $n = 20$ samples for noise attenuation. Regarding the XAI confidence scores synthesis strategy, we assess all three alternatives, namely heatmap average (HA), heatmap derivative (HD), and heatmap reconstruction loss (HRL). We also vary the windowing method strategy in the detection window, using mean or max. On the detection sequences, if the mean/max score is higher than the automatically estimated threshold γ_{95}, an alarm is triggered (see Section 4.1.3).

We executed ThirdEye to capture the attention maps for all simulations of our evaluation set using the studied ADS as input. For the external unknown scenarios ($\text{OOD}_{\text{extreme}}$ and $\text{OOD}_{\text{moderate}}$), we used the DAVE-2 model from the replication package of our baseline [60]. For the internal uncertain scenarios (Mutants), we used all the selected 20 mutants.

5.4.4 Baseline. We use SelfOracle [60], a black-box misbehaviour predictor, as baseline for ThirdEye. We chose SelfOracle for the following reasons: (1) it is designed for the task of failure prediction of ADS; (2) it is a competitive approach; results show that it outperforms the input validation strategy of DeepRoad [77]; (3) it was developed, integrated, and experimented on the Udacity simulator, which mitigates the threats to the internal validity that are possible when experimenting a tool in a simulation environment different from the one in which it was implemented.

We use the best configuration of SelfOracle presented in the original paper, i.e., a variational autoencoder (VAE) that reconstructs *driving images* and uses the reconstruction loss as a measure of confidence. The VAE has a latent size of 2 and it was trained to minimize the MSE (see Section 2.3) between the original and reconstructed nominal images (sunny). In the original SelfOracle paper, only the arithmetic mean of the detection window was evaluated [60]. In our study, we evaluate two configurations of SelfOracle, using both the mean and the max computed on the detection window, the latter being a new experimental contribution of this work.

5.4.5 Metrics used for Analysis. We compute the *true positives* as the number of correct failure predictions within the predefined TTF (see Section 5.4.2) and the *false negatives* as the number of missed failure predictions when our framework does not trigger an alarm in a detection window. We remember that the false positives and true negatives are measured using nominal simulations.

Our primary goal is to achieve high Recall (Re), or true positive rate, defined as Re=TP/(TP+FN)). Recall measures the fraction of safety-critical failures detected by a technique. It is also important to achieve high precision (Pr), defined as Pr=TP/(TP+FP). Precision measures the fraction of correct warnings that a technique reports. We also consider the F_{beta} score [8], with $\beta = 3.0$, as a weighted balance between precision and recall ($F_3 = \frac{10 \cdot \text{Precision} \times \text{Recall}}{9 \cdot \text{Precision} + \text{Recall}}$). We are interested in an F-measure that weights recall higher compared to precision, because the cost associated with false negatives is very high in the safety-critical domain [8] as it means a missed failure detection. In contrast, in our setting, the cost associated with false positives is relatively lower compared to false negatives. A false alarm causes annoyance to the human driver (or to the ADS) when there is no actual hazard; thus their number should be kept low.

5.5 Results

5.5.1 Effectiveness (RQ$_1$). Table 1 presents the effectiveness results for all configurations of ThirdEye (HA, HD, HRL) and SelfOracle (Rec. Loss), divided by windowing function (max, mean). Results are averaged across conditions, split between external unknown conditions ($\text{OOD}_{\text{extreme}}$ and $\text{OOD}_{\text{moderate}}$) and internal uncertain conditions (Mutants). The effectiveness metrics consider a confidence threshold γ_{95} (see Section 5.4.3), i.e., the threshold associated with an expected 5% false positive rate. Precision (Pr) is measured in anomalous conditions, which explains why it is lower than the expected value associated with γ_{95}, which is 95% in nominal conditions (see Section 4.1.3).

Due to space constraints, in this section, we only comment the average F_3 scores over all benchmarks. On average, in terms of F_3, ThirdEye with windowing=max has a 77%/39%/60% average improvement over ThirdEye with windowing=mean, for HA/HD/HRL, respectively. However, this does not impact negatively the false alarm rate, which remains low (average Pr values for windowing=max are higher than those for windowing=mean). HD/HRL scores are slightly higher in terms of F_3 than HA (+2%).

We assessed the statistical significance of these differences using the non-parametric Mann-Whitney U test [72] (with $\alpha = 0.05$) and the magnitude of the differences using the Cohen's d effect size [15]. The difference in F_3 score between HA and HD/HRL were found to be statistically significant (p-value < 0.05) even if with a negligible and small effect sizes. As expected by looking at the average F_3 scores of Table 1, there is no statistically significant difference between HD and HRL (p-value ≥ 0.05).

RQ$_1$: *The configuration of ThirdEye using heatmap derivative function (HD) and reconstruction loss function (HRL), configured with windowing=max, achieve the highest failure prediction rate (F_3 = 85%) over all conditions.*

Table 1: Results for all failure predictors. Scores are computed for γ_{95}. Average F_3 scores are highlighted in bold, best F_3 scores are highlighted in grey.

Benchmark		ThirdEye									SelfOracle			ThirdEye									SelfOracle		
		Windowing: max												Windowing: mean											
		HA			HD			HRL			Rec. Loss			HA			HD			HRL			Rec. Loss		
	TTF (s)	Pr	Re	F_3	Pr	Re	F_3	Pr	Re	F_3	Pr	Re	F_3	Pr	Re	F_3	Pr	Re	F_3	Pr	Re	F_3	Pr	Re	F_3
$\text{OOD}_\text{extreme}$	1	40	90	79	30	100	80	27	86	76	24	90	33	26	46	36	19	58	54	10	25	25	20	79	33
	2	37	86	81	29	100	80	26	86	79	24	92	32	32	60	55	21	64	55	14	38	32	16	68	32
	3	41	93	82	30	100	80	29	93	82	20	77	33	38	71	65	24	71	62	21	51	47	16	68	33
	avg	39	90	**81**	30	100	**80**	27	89	**79**	23	87	**33**	32	59	**52**	21	64	**57**	15	38	**35**	17	71	**33**
$\text{OOD}_\text{moderate}$	1	49	92	84	35	99	83	35	100	83	24	100	75	8	17	15	27	65	56	26	79	65	24	100	75
	2	41	72	66	33	96	80	34	100	83	23	100	72	25	41	39	22	46	40	24	76	62	22	92	65
	3	41	71	66	33	94	78	34	95	79	23	93	59	24	36	34	25	49	44	24	72	59	18	72	45
	avg	44	78	**72**	34	96	**80**	34	98	**82**	23	98	**69**	19	31	**29**	25	53	**47**	25	76	**62**	21	88	**62**
Mutants	1	82	100	98	70	100	96	70	100	96	57	97	90	66	61	61	75	99	95	61	88	84	41	59	56
	2	82	100	98	70	99	95	70	100	96	47	70	66	66	56	57	70	80	78	50	65	62	6	9	9
	3	81	100	98	69	100	96	69	100	96	40	54	52	68	59	59	64	67	66	41	42	42	1	3	2
	avg	82	100	**98**	69	100	**96**	69	100	**96**	48	74	**69**	67	58	**59**	70	82	**80**	51	65	**63**	16	24	**22**
Average (All)	1	57	94	87	45	100	86	44	95	85	35	96	66	33	41	37	40	74	68	32	64	58	28	79	55
	2	53	86	82	44	99	85	43	95	86	31	87	57	41	52	50	37	63	58	29	60	52	15	56	35
	3	54	88	82	44	98	85	44	96	85	27	75	48	43	55	53	37	63	58	29	55	49	12	48	27
	avg	55	89	**83**	44	99	**85**	44	96	**85**	31	86	**57**	39	50	**47**	38	67	**61**	30	60	**53**	18	61	**39**

5.5.2 Prediction Over Time (RQ$_2$). Table 1 reports the effectiveness considering different TTF (Column 2). In principle, failure prediction should get more challenging as we move farther from the failure instant. This is true for all configuration of ThirdEye, except for HA (windowing=mean), in which the average prediction power (F_3) is higher for {2, 3} seconds before the failures that 1 second before them, on average (+30%). For ThirdEye HRL/HD/HA with windowing=max, the F_3 scores remain stable over time. On average, the prediction power decreases only by -6%/-1%/-1% as we move away from the failures.

RQ$_2$: *On average, the effectiveness of the best configurations of ThirdEye (HRL/HD windowing=max) remains high up to 3 seconds before the failures (-1% average F_3 decrease).*

5.5.3 Comparison (RQ$_3$). Considering the average F_3 scores across benchmarks, all configurations of ThirdEye are superior to SelfOracle at predicting misbehaviours.

On the $\text{OOD}_\text{extreme}$ benchmark, ThirdEye scores a +142% increase in F_3 w.r.t. SelfOracle (the benchmark used in that work). For $\text{OOD}_\text{moderate}$ conditions, average F_3 scores raise to 62%, for ThirdEye’s windowing=max, whereas the failure detection rate by ThirdEye (HD) is +16% higher (80%). For Mutants, our results show a remarkable difference of effectiveness between ThirdEye over SelfOracle. The configuration HD (windowing=max) predicts *all failures* induced by internal uncertain conditions (Re=100%), a +35% increase w.r.t SelfOracle, whereas for F_3 the increment is +39%.

Overall, average results for F_3 show significant improvements of ThirdEye over SelfOracle, regardless of the configuration being used and the reaction period considered. The best configurations HD/HRL (windowing=max) from RQ$_1$ achieve +49% failure prediction scores (F_3). We assess the statistical significance of the differences between ThirdEye HD and SelfOracle using the non-parametric Mann-Whitney U test [72] (with $\alpha = 0.05$), the magnitude of the differences using the Cohen’s d effect size [15]. Statistical tests tell that the difference in F3 score between HD are statistically significant (p-value < 0.05) with a large effect size.

RQ$_3$: *ThirdEye outperforms SelfOracle in terms of failure prediction and minimization of false alarms (see F_3), with statistical significance.*

We analyzed qualitatively some of the failures, to understand the reasons behind the disagreements between ThirdEye and SelfOracle. Figure 6 reports a meaningful example from our experiments concerning a failure induced by mutation testing. In the example (not show in Figure 6 for space reasons), a mutated version of DAVE-2 is driving on nominal scenarios and fails in proximity of a bend on the right, missing the bend and proceeding straight off-road.

From the plot on the left, we can see that the white-box approach by ThirdEye (HD windowing=max) is able to anticipate the failure as the XAI confidence score spikes a few seconds before the off-road episode. On the other hand, SelfOracle misses the failure. The reconstruction errors (plot on the right) of the driving frames raise above the threshold only when the ADS is already off-track and

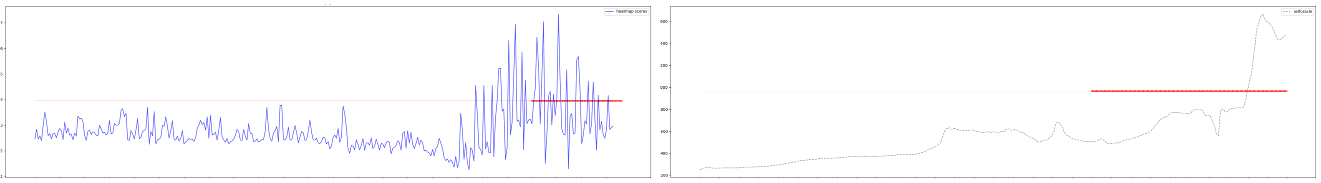

Figure 6: Failure (red stroke) induced by mutation testing; threshold γ is represented by the red line. Left: ThirdEye predicts the failure a few seconds ahead (see spikes above the threshold). Right: SelfOracle reacts only when the failure has happened, when the ADS is off-road, and the input image deviates substantially from the nominal, on-road, driving frames.

the image captured by the camera deviates substantially from the nominal or nearly-failing on-road images.

5.6 Threats to Validity

5.6.1 Internal validity. We compared all variants of ThirdEye and SelfOracle under identical experimental settings and on the same evaluation set (Section 5.4.1). The main threat to internal validity concerns our implementation of the testing scripts to evaluate the failure prediction scores, which we tested thoroughly. Concerning the training of ADS model, we used artifacts publicly available in the replication packages of the SelfOracle [60] and DeepCrime [27] papers. Regarding the simulation platform, we used the Udacity simulator adopted in analogous failure prediction studies [28, 60].

5.6.2 External validity. The limited number of self-driving systems in our evaluation poses a threat in terms of generalizability of our results to other ADS. Moreover, results may not generalize, or generalize differently, when considering other simulation platforms than Udacity. For the attention maps, we considered only attention maps produced by SmoothGrad [55], and the effectiveness of our tool may change when considering different XAI algorithms.

5.6.3 Reproducibility. All our results, the source code of ThirdEye, the simulator, and all subjects are available [65].

6 DISCUSSION

6.1 XAI for Failure Prediction

Our study highlights the complexity and the variety of failure scenarios that runtime monitoring techniques should aim to handle. Attention maps are typically used qualitatively by humans to understand how a DNN processes its inputs. In this paper, we used them quantitatively, under the assumption that they contain information that can potentially be used to assess the behaviour of DNNs [51, 64] and, by extension, of the ADS that use them.

Our approach depends on the capability of attention maps to constitute a reference model of normal driving behaviour. Well-trained DNNs better capture the relevant structures in an image, thus produce more meaningful attention maps than poorly trained DNNs, which rather rely on global image statistics. Apart from this requirement, attention maps offer a more transparent and effective assessment of the ADS behaviour than a black-box technique because they indicate the degree of attention (or lack thereof) of the ADS in response to an input. Our results confirm that they are generally more effective than a competing black-box technique on both unknown (out-of-distribution) and uncertain (mutation testing) scenarios.

6.2 Discussing ThirdEye's Configurations

All configurations of ThirdEye are stable in terms of prediction power and we observed no big drop as we move to a longer duration between prediction and failure. For example, for ADS models produced by automated mutation testing, this can be explained by the fact that these self-driving cars are always characterized by a relatively high proportion of uncertainty internal to the system, which ultimately causes a failure that ThirdEye is able to detect because its predictions are made based on information that reflects the (buggy) internal state of the system.

We evaluated two windowing alternatives, *max* vs *mean*. Each has pros and cons: max is more reactive than mean, as it is enough to observe a spike in the window to trigger an alarm, which may potentially lead to a higher recall. At the same time, usage of max during threshold estimation makes γ_{95} higher, because a higher threshold must be chosen to ensure as few as 5% false positives in nominal conditions. A higher γ_{95} leads naturally to a lower recall. The combination of the two factors, higher reactivity and higher γ_{95} threshold, may either lead to better or to worse performance of max vs mean. Hence, the choice can only be made empirically and it is quite interesting that our empirical results show very clearly and neatly the superiority of max over mean.

6.3 Comparison with Other Approaches

Attention maps by XAI are not the only way to analyze the internal functioning of an ADS. For instance, other white-box approaches have been proposed in the literature, such as solutions based on activation traces [34], cross-layer dissection [69], or uncertainty quantification measures [42]. These methods have two main drawbacks that hinder their applicability as online failure predictors. First, in general, these methods are known for being computationally very expensive, thus they may not be real-time viable solutions. Second, these techniques must be integrated into the development process from the very beginning, as they require a white-box access to the model's architecture and to the training data, because the ADS must be retrained or modified to enable the computation of white-box confidence scores. Unlike these methods, we experiment with attention maps because they do offer a white-box view of the DNN internals without requiring access to the training data, nor the need to modify or retrain the ADS model.

7 RELATED WORK

Identifying unexpected driving scenarios is the number one need during ADS testing, according to the survey with developers and domain experts by Lou et al. [41]. The problem has been tackled by researchers either by (1) generating test cases for ADS, (2) proposing anomaly detection tools. We also provide an overview of (3) generic OOD detectors and (4) the main XAI methods used for ADS testing.

7.1 Test Generation for Autonomous Driving

Test generation techniques mostly use search-based techniques to automatically construct test cases for DNN-based ADS [1, 4, 5, 34, 43, 46, 49, 62, 77]. Test cases are real-world images of driving scenes, or road abstractions that are rendered within a driving simulator. Abdessalem et al. [1, 4, 5] combine genetic algorithms and machine learning to test a pedestrian detection system. Mullins et al. [45] use Gaussian processes to drive the search-based test generation towards yet unexplored regions of the input space, whereas Gambi et al. [21] propose search-based test generation for ADS based on procedural content generation.

Test generators aim to maximize the number of failures, whereas our goal is to predict failures in online mode before they happen. Nevertheless, test generators can be used in conjunction with ThirdEye, to generate conditions for our approach to predict.

7.2 Anomaly Detection in Autonomous Driving

We already discussed SelfOracle [60], for which we performed an explicit empirical comparison in this work. DeepGuard [28] uses the reconstruction error by VAEs to prevent collisions of vehicles with the roadside. DeepRoad [77] validates single driving images based on the distance to the training set, using embeddings rooted in the features extracted by VGGNet. In other works [58, 59], continual learning is used to minimize the false positives of a black-box failure predictor. Hell et al. [24] evaluate three different OOD detection methods, namely VAEs, Likelihood Regret and the generative modelling SSD, for ADS testing on the CARLA simulator. Henriksson et al. [25] use the the negative of the log likelihood as a black-box anomaly score. Borg et al. [12] propose to pair OOD detection with VAEs with object detection for an automated emergency braking system. Strickland et al. [61] use an LSTM solution with multiple metrics to predict collisions with vehicles at crossroads.

Our approach differs from the aforementioned black-box approaches because it uses a white-box confidence score of the system synthesized from the attention maps given by an XAI algorithm. For a broad overview of anomaly detection techniques in autonomous driving, we refer the reader to the survey by Bogdoll et. al [9].

7.3 Generic OOD Detectors

AutoTrainer [78] monitors the training process of a DNN and automatically repairs it when the metrics used during training degrade. In contrast, ThirdEye operates at testing time, in production, to recognize unexpected execution conditions, while AutoTrainer operates at training time to fix common training faults.

Zhang et al. [79] introduce the notion of relative activation and deactivation to interpret the decision behavior of a DNN and propose an algorithm for automatic detection of OOD inputs. The abstraction relies on classifying neurons into different states based on stronger relative selectivity, which is a quantitative method of measuring the impact of a particular neuron, or a subset of them, on the inference of the model as a whole.

The use of this technique raises some challenges, such as which and how many layers should be selected, and how the different layers should be aggregated. SelfChecker [73] is a tool that helps answer these questions, but the evaluation of the DNN prediction is performed for individual inputs. ThirdEye uses attention map analysis and does not require to dissect the model layers or the activation states of the neurons, but just to retrieve the gradients during a normal feedforward pass, making it computationally more efficient and easier to integrate into the ADS development process.

Finally, Suraksha [80] is an automated ADS safety evaluation framework that quantitatively analyzes the safety sensitivity of different versions of an ADS. ThirdEye can be used as one of the safety quantification metrics of Suraksha and help improve the safety parameters of ADS.

7.4 XAI for Autonomous Driving Testing

With the increasing application of DNNs to safety-critical domains such as autonomous during, XAI algorithms represent one of the standard choices to debug DNN's predictions and failures (e.g., during an incident). Moreover, XAI is also being adopted to build novel testing solutions to test DNN-based systems, including, but not limited to, ADS. In this section we focus on the main related propositions, i.e., techniques that use attention maps for ADS testing, and techniques that use XAI as a building block for DNN testing. For a complete overview of the state of the art on XAI for ADS, we refer the reader to the survey by Atakishiyev et al. [3].

VisualBackProp [10] was created to visualize which group of pixels of the input image contributes more to the predictions of a convolutional neural network (CNN). Kim and Canny [33] explore the use of attention maps for explaining the CNN behaviour in a ADS. Lateef et al. [38] uses generative adversarial networks to train a predictive model that generates attention maps from road scenes and gives more prominence to the objects in the scene that are most important to the driver's decision making (e.g., other cars, pedestrians, and traffic lights/signs). Xu et al. [74] investigated the use of XAI techniques to detect action-inducing objects, i.e., objects that have a relevant effect on a driving decision, and jointly predict actions and their respective explanations. Mohseni et al. [44] train the DAVE-2 model to predict a steering angle given the attention maps by VisualBackProp [10]. Similarly, ThirdEye also leverages XAI to increase the level of reliability of an ADS. Differently to the aforementioned works, ThirdEye focuses on failure prediction of lane-keeping based ADS during external unknown and internal uncertain driving conditions.

Fahmy et al. [19] apply clustering to LRP heatmaps capturing the relevance of the DNN predictions to automatically support the identification of failure-inducing inputs. Such data is used for the retraining of a gaze detection system that uses DNNs to determine the gaze direction of the driver. The authors present an extension of the previous work [18] in which inputs identified by the heatmap-based mechanism are given in input to a search-based test generator. In contrast, in this work we use attention maps from SmoothGrad to support the prompt detection of low-confidence scenarios of a lane-keeping DNN-enabled ADS. Zohdinasab et al. [82] use illumination search to cover a feature map of external behaviours of an ADS. These feature maps are used as an adequacy criteria of the inputs generated by an ADS test generator, whereas we use attention maps from the XAI domain to validate the inputs processed by the ADS.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we describe and evaluate a white-box failure predictor that estimates the confidence of a DNN-based ADS in response to unpredictable execution contexts. Our tool ThirdEye performs confidence estimation by turning attention maps derived from the XAI domain into confidence scores of the driving ADS. Our approach is able to anticipate many potentially safety-critical failures by several seconds, with a low false alarm rate in anomalous conditions, and a fixed 5% expected false alarm rate in nominal conditions, outperforming a black-box predictor from the literature.

Future work includes extending the comparison to other white-box confidence estimators. At the same time, alternative confidence score synthesis methods based on the semantic of the input image will be investigated, as well as other XAI algorithms. Moreover, we also plan to extend the detection of finer-grained driving quality degradations (e.g., erratic driving behaviour) and to study self-healing mechanisms within the simulator.

ACKNOWLEDGMENTS

This work was partially supported by the H2020 project PRECRIME, funded under the ERC Advanced Grant 2017 Program (ERC Grant Agreement n. 787703).

REFERENCES

[1] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2018. Testing Autonomous Cars for Feature Interaction Failures Using Many-objective Search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018)*. ACM.

[2] Jinwon An and Sungzoon Cho. 2015. Variational Autoencoder based Anomaly Detection using Reconstruction Probability.

[3] Shahin Atakishiyev, Mohammad Salameh, Hengshuai Yao, and Randy Goebel. 2021. Explainable artificial intelligence for autonomous driving: An overview and guide for future research directions. https://doi.org/10.48550/ARXIV.2112.11561

[4] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. 2016. Testing advanced driver assistance systems using multi-objective search and neural networks. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*.

[5] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. 2018. Testing Vision-Based Control Systems Using Learnable Evolutionary Algorithms. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*.

[6] BGR Media, LLC. 2018. Waymo's self-driving cars hit 10 million miles. https://techcrunch.com/2018/10/10/waymos-self-driving-cars-hit-10-million-miles. Online; accessed 18 August 2019.

[7] Alexander Binder, Grégoire Montavon, Sebastian Bach, Klaus-Robert Müller, and Wojciech Samek. 2016. Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers. *CoRR* abs/1604.00825 (2016). arXiv:1604.00825 http://arxiv.org/abs/1604.00825

[8] David C. Blair. 1979. Information Retrieval, 2nd ed. C.J. Van Rijsbergen. London: Butterworths; 1979: 208 pp. Price: $32.50. *Journal of the American Society for Information Science* 30, 6 (1979), 374–375. https://doi.org/10.1002/asi.4630300621 arXiv:https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.4630300621

[9] Daniel Bogdoll, Maximilian Nitsche, and J. Marius Zöllner. 2022. Anomaly Detection in Autonomous Driving: A Survey. https://doi.org/10.48550/ARXIV.2204.07974

[10] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Larry Jackel, Urs Muller, and Karol Zieba. 2016. VisualBackProp: efficient visualization of CNNs. https://doi.org/10.48550/ARXIV.1611.05418

[11] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* abs/1604.07316 (2016).

[12] Markus Borg, Jens Henriksson, Kasper Socha, Olof Lennartsson, Elias Sonnsjö Lönegren, Thanh Bui, Piotr Tomaszewski, Sankar Raman Sathyamoorthy, Sebastian Brink, and Mahshid Helali Moghadam. 2022. Ergo, SMIRK is Safe: A Safety Case for a Machine Learning Component in a Pedestrian Automatic Emergency Brake System. https://doi.org/10.48550/ARXIV.2204.07874

[13] Vinton G. Cerf. 2018. A Comprehensive Self-driving Car Test. *Commun. ACM* 61, 2 (Jan. 2018).

[14] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009).

[15] Jacob Cohen. 1988. *Statistical power analysis for the behavioral sciences*. L. Erlbaum Associates, Hillsdale, N.J.

[16] Swaroopa Dola, Matthew B Dwyer, and Mary Lou Soffa. 2021. Distribution-aware testing of neural networks using generative models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 226–237.

[17] Brian S. Everitt, Sabine Landau, and Morven Leese. 2009. *Cluster Analysis* (4th ed.). Wiley Publishing.

[18] Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. 2022. Simulator-based explanation and debugging of hazard-triggering events in DNN-based safety-critical systems. https://doi.org/10.48550/ARXIV.2204.00480

[19] Hazem M. Fahmy, Mojtaba Bagherzadeh, Fabrizio Pastore, and Lionel C. Briand. 2020. Supporting DNN Safety Analysis and Retraining through Heatmap-based Unsupervised Learning. *CoRR* abs/2002.00863 (2020). arXiv:2002.00863 https://arxiv.org/abs/2002.00863

[20] Yarin Gal and Zoubin Ghahramani. 2016. Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org.

[21] Alessio Gambi, Marc Mueller, and Gordon Fraser. 2019. Automatically Testing Self-driving Cars with Search-based Procedural Content Generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019)*. ACM.

[22] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2020. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics* 37, 3 (2020), 362–386.

[23] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel Briand. 2020. 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 (ICST '20)*. IEEE.

[24] Franz Hell, Gereon Hinz, Feng Liu, Sakshi Goyal, Ke Pei, Tetiana Lytvynenko, Alois Knoll, and Chen Yiqiang. 2021. Monitoring Perception Reliability in Autonomous Driving: Distributional Shift Detection for Estimating the Impact of Input Data on Prediction Accuracy. In *Computer Science in Cars Symposium* (Ingolstadt, Germany) *(CSCS '21)*. Association for Computing Machinery, New York, NY, USA, Article 8, 9 pages. https://doi.org/10.1145/3488904.3493382

[25] Jens Henriksson, Christian Berger, Markus Borg, Lars Tornberg, Cristofer Englund, Sankar Raman Sathyamoorthy, and Stig Ursing. 2019. Towards Structured Evaluation of Deep Neural Network Supervisors. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE.

[26] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of Real Faults in Deep Learning Systems. In *Proceedings of 42nd International Conference on Software Engineering (ICSE '20)*. ACM, 12 pages.

[27] Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. 2021. DeepCrime: Mutation Testing of Deep Learning Systems Based on Real Faults. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual, Denmark) *(ISSTA 2021)*. Association for Computing Machinery, New York, NY, USA, 67–78. https://doi.org/10.1145/3460319.3464825

[28] Manzoor Hussain, Nazakat Ali, and Jang-Eui Hong. 2022. DeepGuard: A Framework for Safeguarding Autonomous Driving Systems from Inconsistent Behaviour. *Automated Software Engg.* 29, 1 (may 2022), 32 pages. https://doi.org/10.1007/s10515-021-00310-0

[29] Tech. Rep. ISO/PAS 21448:2019 International Organization for Standardization. 2019. Road Vehicles - Safety of the Intended Functionality.

[30] Gunel Jahangirova, Andrea Stocco, and Paolo Tonella. 2021. Quality Metrics and Oracles for Autonomous Vehicles Testing. In *Proceedings of 14th IEEE International Conference on Software Testing, Verification and Validation (ICST '21)*. IEEE.

[31] Saumya Jetley, Nicholas A Lord, Namhoon Lee, and Philip HS Torr. 2018. Learn to pay attention. *arXiv preprint arXiv:1804.02391* (2018).

[32] Yeon-Jee Jung, Seung-Ho Han, and Ho-Jin Choi. 2021. Explaining CNN and RNN Using Selective Layer-Wise Relevance Propagation. *IEEE Access* 9 (2021), 18670–18681. https://doi.org/10.1109/ACCESS.2021.3051171

[33] Jinkyu Kim and John Canny. 2017. Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention. https://doi.org/10.48550/ARXIV.1703.10631

[34] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing Using Surprise Adequacy. In *Proceedings of the 41st International Conference on Software Engineering (ICSE '19)*. IEEE Press.

[35] Teuvo Kohonen. 2001. *Self-Organizing Maps, Third Edition.* Springer.

[36] Yasuhiro Kubota. 2021. *tf-keras-vis.* https://keisen.github.io/tf-keras-vis-docs/

[37] Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2019. An Evaluation of the Human-Interpretability of Explanation. *CoRR* abs/1902.00006 (2019). arXiv:1902.00006 http://arxiv.org/abs/1902.00006

[38] Fahad Lateef, Mohamed Kas, and Yassine Ruichek. 2021. Saliency Heat-Map as Visual Attention for Autonomous Driving Using Generative Adversarial Network (GAN). *IEEE Transactions on Intelligent Transportation Systems* (2021), 1–14. https://doi.org/10.1109/TITS.2021.3053178

[39] Jeong Keun Lee and Kang Wook Lee. 2013. Study on Effectiveness of Pre-Crash Active Seatbelt Using Real Time Controlled Simulation.

[40] Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. Rationalizing Neural Predictions. *CoRR* abs/1606.04155 (2016). arXiv:1606.04155 http://arxiv.org/abs/1606.04155

[41] Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. 2021. Investigation into the state-of-the-practice autonomous driving testing. https://doi.org/10.48550/ARXIV.2106.12233

[42] Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. 2020. Uncertainty Quantification with Statistical Guarantees in End-to-End Autonomous Driving Control. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020.* IEEE, 7344–7350. https://doi.org/10.1109/ICRA40945.2020.9196844

[43] Mahshid Helali Moghadam, Markus Borg, Mehrdad Saadatmand, Seyed Jalaleddin Mousavirad, Markus Bohlin, and Björn Lisper. 2022. Machine Learning Testing in an ADAS Case Study Using Simulation-Integrated Bio-Inspired Search-Based Testing. https://doi.org/10.48550/ARXIV.2203.12026

[44] Sina Mohseni, Akshay Jagadeesh, and Zhangyang Wang. 2019. Predicting Model Failure using Saliency Maps in Autonomous Driving Systems. *CoRR* abs/1905.07679 (2019). arXiv:1905.07679 http://arxiv.org/abs/1905.07679

[45] Galen E. Mullins, Paul G. Stankiewicz, R. Chad Hawthorne, and Satyandra K. Gupta. 2018. Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. *Journal of Systems and Software* 137 (2018).

[46] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17).* ACM.

[47] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 1135–1144.

[48] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing Machine Learning based Systems: A Systematic Mapping. *Empirical Software Engineering* (2020).

[49] Vincenzo Riccio and Paolo Tonella. 2020. Model-Based Exploration of the Frontier of Behaviours for Deep Learning System Testing. In *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20).*

[50] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. 2019. *Explainable AI: interpreting, explaining and visualizing deep learning.* Vol. 11700. Springer Nature.

[51] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296* (2017).

[52] Bernhard Schölkopf, Robert C. Williamson, Alexander J. Smola, John Shawe-Taylor, and John C. Platt. 1999. Support Vector Method for Novelty Detection. In *Advances in Neural Information Processing Systems 12, (NIPS).*

[53] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc.

[54] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR* abs/1610.02391 (2016). arXiv:1610.02391 http://arxiv.org/abs/1610.02391

[55] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. 2017. SmoothGrad: removing noise by adding noise. *CoRR* abs/1706.03825 (2017). arXiv:1706.03825 http://arxiv.org/abs/1706.03825

[56] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806 [cs.LG]

[57] Andrea Stocco, Brian Pulfer, and Paolo Tonella. 2022. Mind the Gap! A Study on the Transferability of Virtual vs Physical-world Testing of Autonomous Driving Systems. *IEEE Transactions on Software Engineering* (2022). https://doi.org/10.1109/TSE.2022.3202311

[58] Andrea Stocco and Paolo Tonella. 2020. Towards Anomaly Detectors that Learn Continuously. In *Proceedings of 31st International Symposium on Software Reliability Engineering Workshops (ISSREW 2020).* IEEE.

[59] Andrea Stocco and Paolo Tonella. 2021. Confidence-driven Weighted Retraining for Predicting Safety-Critical Failures in Autonomous Driving Systems. *Journal of Software: Evolution and Process* (2021). https://doi.org/10.1002/smr.2386

[60] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. 2020. Misbehaviour Prediction for Autonomous Driving Systems. In *Proceedings of 42nd International Conference on Software Engineering (ICSE '20).* ACM.

[61] Mark Strickland, Georgios Fainekos, and Hani Ben Amor. 2018. Deep predictive models for collision risk assessment in autonomous driving. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018 (Proceedings - IEEE International Conference on Robotics and Automation).* Institute of Electrical and Electronics Engineers Inc.

[62] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18).* ACM.

[63] Erico Tjoa and Cuntai Guan. 2019. A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI. *CoRR* abs/1907.07374 (2019). arXiv:1907.07374 http://arxiv.org/abs/1907.07374

[64] Erico Tjoa, Hong Jing Khok, Tushar Chouhan, and Cuntai Guan. 2022. Improving Deep Neural Network Classification Confidence using Heatmap-based eXplainable AI. *CoRR* abs/2201.00009 (2022). arXiv:2201.00009 https://arxiv.org/abs/2201.00009

[65] ThirdEye 2022. Replication Package. https://github.com/tsigalko18/ase22.

[66] Udacity. 2017. A self-driving car simulator built with Unity. https://github.com/udacity/self-driving-car-sim. Online; accessed 18 August 2019.

[67] unity 2021. Unity3D. https://unity.com.

[68] National Highway Traffic Safety Administration U.S. Department of Transportation. 2007. Pre-Crash Scenario Typology for Crash Avoidance Research.

[69] Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, and Jian Lu. 2020. DISSECTOR: Input Validation for Deep Learning Applications by Crossing-layer Dissection. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE).* 727–738.

[70] Waymo Driver 2021. Waymo Driver. https://waymo.com/waymo-driver/.

[71] waymos-secret-testing 2017. Waymo Secret Testing. https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

[72] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (Dec. 1945), 80. https://doi.org/10.2307/3001968

[73] Yan Xiao, Ivan Beschastnikh, David S. Rosenblum, Changsheng Sun, Sebastian Elbaum, Yun Lin, and Jin Song Dong. 2021. Self-Checking Deep Neural Networks in Deployment. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE).* 372–384. https://doi.org/10.1109/ICSE43902.2021.00044

[74] Yiran Xu, Xiaoyin Yang, Lihang Gong, Hsuan-Chu Lin, Tz-Ying Wu, Yunsheng Li, and Nuno Vasconcelos. 2020. Explainable Object-induced Action Decision for Autonomous Vehicles. https://doi.org/10.48550/ARXIV.2003.09405

[75] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access* 8 (2020), 58443–58469.

[76] Matthew D. Zeiler and Rob Fergus. 2013. Visualizing and Understanding Convolutional Networks. *CoRR* abs/1311.2901 (2013). arXiv:1311.2901 http://arxiv.org/abs/1311.2901

[77] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018).* ACM.

[78] Xiaoyu Zhang, Juan Zhai, Shiqing Ma, and Chao Shen. 2021. AUTOTRAINER: An Automatic DNN Training Problem Detection and Repair System. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE).* 359–371. https://doi.org/10.1109/ICSE43902.2021.00043

[79] Zhen Zhang, Peng Wu, Yuhang Chen, and Jing Su. 2021. Out-of-Distribution Detection through Relative Activation-Deactivation Abstractions. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE).* 150–161. https://doi.org/10.1109/ISSRE52982.2021.00027

[80] Hengyu Zhao, Siva Kumar Sastry Hari, Timothy Tsai, Michael B. Sullivan, Stephen W. Keckler, and Jishen Zhao. 2021. Suraksha: A Framework to Analyze the Safety Implications of Perception Design Choices in AVs. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE).* 434–445. https://doi.org/10.1109/ISSRE52982.2021.00052

[81] Zhiguo Zhao, Liangjie Zhou, Qiang Zhu, Yugong Luo, and Keqiang Li. 2017. A review of essential technologies for collision avoidance assistance systems. *Advances in Mechanical Engineering* 9, 10 (2017).

[82] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. DeepHyperion: Exploring the Feature Space of Deep Learning-Based Systems through Illumination Search. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21).* Association for Computing Machinery.