# Predicting Safety Misbehaviours in Autonomous Driving Systems using Uncertainty Quantification

Ruben Grewal
*Technical University of Munich*
Munich, Germany
ruben.grewal@tum.de

Paolo Tonella
*Software Institute - USI*
Lugano, Switzerland
paolo.tonella@usi.ch

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

***Abstract*—The automated real-time recognition of unexpected situations plays a crucial role in the safety of autonomous vehicles, especially in unsupported and unpredictable scenarios. This paper evaluates different Bayesian uncertainty quantification methods from the deep learning domain for the anticipatory testing of safety-critical misbehaviours during system-level simulation-based testing. Specifically, we compute uncertainty scores as the vehicle executes, following the intuition that high uncertainty scores are indicative of unsupported runtime conditions that can be used to distinguish safe from failure-inducing driving behaviors. In our study, we conducted an evaluation of the effectiveness and computational overhead associated with two Bayesian uncertainty quantification methods, namely MC-Dropout and Deep Ensembles, for misbehaviour avoidance. Overall, for three benchmarks from the Udacity simulator comprising both out-of-distribution and unsafe conditions introduced via mutation testing, both methods successfully detected a high number of out-of-bounds episodes providing early warnings several seconds in advance, outperforming two state-of-the-art misbehaviour prediction methods based on autoencoders and attention maps in terms of effectiveness and efficiency. Notably, Deep Ensembles detected most misbehaviours without any false alarms and did so even when employing a relatively small number of models, making them computationally feasible for real-time detection. Our findings suggest that incorporating uncertainty quantification methods is a viable approach for building fail-safe mechanisms in deep neural network-based autonomous vehicles.**

***Index Terms*—autonomous vehicles testing, uncertainty quantification, self-driving cars, failure prediction.**

## I. Introduction

Autonomous driving systems (ADS) are vehicles equipped with sensors, cameras, radar, and artificial intelligence, used to let them travel between destinations without human intervention. For a vehicle to be qualified as fully autonomous, it must possess the capability to autonomously navigate to a predefined destination on roads that have not been specifically adapted for its use [1]. The U.S. Department of Transportation, National Highway Traffic Safety Administration (NHTSA), has defined five standardized levels of autonomy, from driver assistance (with the driver being responsible for safe driving) to full automation (where no human driver is required to operate the vehicle). Several companies, such as Audi, BMW, Ford, Google, General Motors, Tesla, Volkswagen, and Volvo, are actively engaged in the development and testing of autonomous vehicles. In recent years, we witnessed advancements such as people hailing self-driving taxis or fleets of fully automated cars with no accompanying safety drivers. Deep neural networks (DNNs) are the driving force behind self-driving car systems. To create autonomous vehicles, developers rely on extensive datasets harnessed in the field to train large DNNs. This data includes images captured by cameras on actual vehicles and other sensors, enabling the DNNs to learn to identify road elements, traffic lights, pedestrians, and other elements within diverse driving environments [2].

Safety assessment of ADS is a hard endeavor and extensive testing is required before deployment on public roads. To validate the safety of ADS, companies adopt a multi-pillar approach that encompasses simulation-based testing, test track, and real-world testing [3], [4]. Researchers have focused primarily on the first pillar, proposing automated testing techniques that try to expose failing conditions and corner cases [5]–[9]. However, despite these efforts, public acceptance of autonomous driving software in the real world would consider the capabilities of the ADS to operate safely in partially unknown and uncertain environments, therefore exhibiting a high level of robustness also for sensor inaccuracies and environmental uncertainties [10].

DNNs are known for their tendency to produce unexpectedly incorrect yet overly confident predictions, particularly in complex environments like autonomous driving. This poses significant safety concerns for ADS, which should possess situational awareness capabilities to discern challenging scenarios, such as adverse weather conditions, which are likely to induce errors and then prompt timely warnings to the driver or trigger fail-safe mechanisms [11], [12].

Previous research has introduced techniques to build safety in-service monitoring [13]–[19]. Frameworks such as SelfOracle [18], DeepRoad [16], DeepGuard [20] require a data-box access [21] and they are capable of analyzing real-world driving data and assess whether the ADS is safe. However, these approaches work in a *black-box* manner (i.e., they analyze the input/output data and identify anomalous instances, without considering the internal processing by the DNN model), which makes them less sensitive to bugs at the model level [22] and prone to false positives/negatives, given their external perspective on the system being tested. A recent

978-1-5386-5541-2/20/$31.00 ©2024 IEEE

white-box solution uses attention maps as a proxy of the DNN uncertainty to enhance the accuracy of failure prediction [23], but it comes with higher costs and therefore is less suitable for resource-constrained environments.

This paper investigates the problem of building a white-box ADS failure predictor rooted in the uncertainty quantification (UQ) methods available in the deep learning domain. Uncertainty quantification consists of approaches that compute the confidence, or lack thereof, of deep learning models in response to certain inputs [11]. UQ is widely used for the analysis, testing, comprehension, and debugging of DNNs [11], [12]. In this work, we evaluate two UQ methods for failure prediction to keep the reliability of the ADS within safety bounds. Our approach leverages uncertainty scores as a transparent confidence estimator for the system. Online monitoring is performed during ADS driving; the uncertainty scores synthesized from the internals of the DNN under test are used to automatically identify conditions in which the system is not confident. In this paper, we show that uncertainty scores represent important clues about the reliability of the ADS and can be used as failure predictors. Our technique works unsupervisedly as failure prediction is performed by establishing a threshold over the uncertainty scores during nominal operating conditions. Hence, anomalous driving conditions are detected when the uncertainty scores increase above such threshold within a specific detection window preceding the failure.

We have evaluated the effectiveness of uncertainty quantification methods on the Udacity simulator for self-driving cars [24], using ADS available from the literature and a diverse set of failures induced by adverse operational scenes and mutation testing-simulated malfunctions. More specifically, we evaluated two uncertainty quantification methods (i.e., Monte Carlo Dropout and Deep Ensembles) and their effectiveness when varying their hyperparameters (e.g., number of models or samples used for uncertainty estimation) at different confidence levels. In our experiments using an existing dataset of +70 simulations accounting for more than 250 failures [23], UQ methods demonstrated remarkable predictive capabilities, forecasting most failures several seconds in advance, a 6-15% increase in failures detected compared to SelfOracle [18] and ThirdEye [23], two state-of-the-art strategies from the literature based on autoencoders and attention maps. Notably, our most successful UQ method strikes a superior balance between identifying misbehaviors and minimizing false alarms (94% $F_3$ score) for a relatively constrained configuration, ensuring computational feasibility for real-time detection.

Our paper makes the following contributions:

**Technique.** A monitoring technique for ADS failure prediction based on uncertainty quantification methods. Our approach is publicly available as a tool [25].

**Evaluation.** An empirical study showing that the uncertainty scores are a promising white-box confidence metric for failure prediction, outperforming the black-box approach of SelfOracle [18] and the XAI-based approach by ThirdEye [23]. Our study also discusses the performance of our methods for real-time prediction.

## II. Background

### A. Lane-keeping ADS

ADS rely on sensor data, cameras, and GPS to perceive their surroundings and use different processing methods to enable predictive decisions regarding vehicle controls [1].

From an architectural point of view, ADS can be mainly divided into two categories: end-to-end ADS driving models and multi-module ADS. The former ones are based on advanced DNNs that are trained on massive datasets of driving scenes. The latter ones are organized into four modules: perception, prediction, planning, and control [1]. The perception module receives as input various sources of sensor data, such as images of the front camera, and proximity sensor, to detect objects in the neighborhood of the vehicle. The prediction module predicts the trajectories of these objects, which are used by the planning module to decide a safe route. The control module translates the route into actual vehicle commands, e.g., a sequence of steering angles. As of now, the two approaches coexist [1] and it is not clear if an approach will prevail.

In this paper, we consider testing end-to-end ADS, while we leave the investigation of multi-module ADS for future work. Particularly, we focus on ADS that implement the "behavioral cloning" task through imitation learning. In this task, the vehicle learns the function of lane-keeping in an end-to-end manner, from human-labeled driving samples in which actuators' values reflect the driving decisions of an expert human driver operating a real physical vehicle, or a simulated vehicle within a driving simulator [24]. Once trained, models like NVIDIA's DAVE-2 [26] are capable of predicting the vehicle's controls (i.e., steer, brake, acceleration).

The ability to keep the vehicle within a lane is a fundamental component of the safe deployment of DNN-based ADS. Notably, the NHTSA has reported that off-road failures are not only frequent but also come at a significant cost, exceeding 15 billion USD [27].

### B. Failure Conditions for Lane-keeping ADS

In the context of NHTSA Level 4 (High Automation), a system monitor plays a critical role in identifying emerging functional insufficiencies. Its primary objective is to maintain a high level of functional quality, even in extreme situations [13], [16], [17]. When the monitor deems the current condition as unsafe, the ADS should be designed to disengage, requesting human intervention to take control of the vehicle, or activating alternative fail-safe mechanisms [12].

Among the underlying causes of ADS failures, such as instances of off-road driving, SOTIF [28] highlights the role of both external unknown and internal uncertain conditions [28]. External unknown conditions encompass "abnormal" inputs that represent rare, unexpected, and potentially unsupported environmental events. These conditions typically involve scenarios where the ADS was not trained due to the absence of prior knowledge (i.e., epistemic uncertainty), such as specific road types or particular weather and lighting conditions. The DNNs utilized within ADS may not be resilient to such

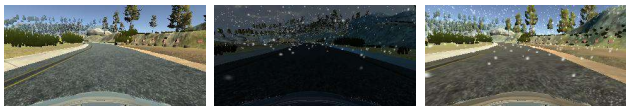

Fig. 1: Examples of operational conditions [23]. Left: nominal (sunny). Center: OOD (night+snow). Right: OOD (snow).

significant changes in data distribution and they are said to be out-of-distribution (OOD, see Figure 1), potentially resulting in system-level failures such as the ADS driving off-road. Conversely, internal uncertain conditions pertain to misbehaviors within the decision component of the ADS. These misbehaviors are often attributed to inherent bugs in the DNN model, which may be introduced during its development phase. Common instances of such bugs include inadequate training data and sub-optimal choices regarding the model architecture or training hyperparameters [22]. In the rest of the paper, we shall use the terms failures/misbehaviours interchangeably.

### C. Existing Unsupervised Failure Predictions Methods

Researchers have proposed ADS failure prediction models that can be trained with no supervision (i.e., no knowledge of the anomalies). Certain propositions are based on a data-box access [21][1] to the main system [13], [18]–[20], whereas other solutions require internal information of the systems and therefore are considered white-box [23], [29], [30].

In this work, we chose two representative propositions from both domains, namely SelfOracle [18] and ThirdEye [23]. We selected these approaches as baselines because they represent two competitive approaches, one black-box, and one white-box, that are designed for the task of failure prediction of ADS and use an unsupervised failure predictor to analyze inputs and assign a suspiciousness score to them, which should be low (below a threshold) if the inputs are supported, or high (above a threshold) otherwise.

These approaches were developed, integrated, and experimented on the Udacity simulator [24]. In this paper, we evaluate our failure predictors in the same experimental setting as previous work to mitigate the threats to the internal validity that are possible when experimenting with tools in a simulation environment different from the one in which they were originally implemented. In the following of this section, we provide further details on the two baseline approaches.

SelfOracle [18] is a black-box technique that estimates the system confidence by analyzing the front-facing camera images used by the ADS. SelfOracle uses an autoencoder to reconstruct driving images and the reconstruction loss as a measure of confidence. The autoencoder 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 unsupported sample. While effective, the main criticism of SelfOracle is that it is not informed by the internal functioning of the DNNs responsible for controlling the ADS, as its only connection with such DNNs is the common training set (i.e., the same inputs are used to train DNNs and autoencoder, which makes these or similar inputs relatively familiar and easy to handle/reconstruct for both DNNs/autoencoder).

To address this, ThirdEye [23] was proposed as a white-box alternative based on the attention maps produced by explainable artificial intelligence techniques (XAI). ThirdEye synthesizes suspiciousness scores using different strategies (i.e., pixel-level average, or autoencoder-based reconstruction loss). While proved promising, such confidence scores are only a proxy of the true uncertainty. Second, computing heatmaps at runtime requires a non-negligible computational overhead, which makes their application as a runtime monitoring prediction system a careful, if at all possible, choice.

In this paper, we aim to ground the benefits of UQ for misbehaviour prediction and compare them with such existing approaches. While uncertainty quantifiers are expected to be informative as they are based on full access to the DNN's internals, they are also known to be computationally expensive. To the best of our knowledge, no empirical comparison has been conducted concerning their effectiveness and efficiency, which represent the core objectives of this work.

## III. Deep Neural Networks Uncertainty Quantification Methods

Uncertainty quantification has gained an increasingly pivotal role in ensuring the reliability and robustness of DNNs, especially those tasked with making critical decisions. Uncertainty can be classified into two main types: aleatoric uncertainty and epistemic uncertainty [31]. Aleatoric uncertainty arises because of the random nature of the system under study, while epistemic uncertainty stems from the lack of knowledge of the system. Aleatoric uncertainty cannot be reduced but can be identified and quantified. Conversely, epistemic uncertainty can be reduced through methods such as sensitivity analysis [32], re-training, and fine-tuning. The total predictive uncertainty can be regarded as the sum of aleatoric and epistemic uncertainty [11].

In the following, we summarize two popular UQ methods proposed in the literature, namely Monte Carlo dropout and Deep Ensembles, and their significance in supervising regression DNNs, such as the ones employed for ADS [12], [33]

### A. Monte Carlo Dropout

The first considered UQ method is Monte Carlo Dropout (MC-Dropout or MDC for short) [34]. In DNNs, dropout layers are used at training time as a regularization method to avoid overfitting. At testing time they are usually disabled for efficiency reasons, and the final DNN prediction would be deterministic. However, uncertainty-aware DNNs based on

[1]In the original papers, these solutions are described as black-box methods, despite their reliance on access to the training set of the ADS. Therefore, it would be more accurate to consider them as data-box techniques. However, for the sake of simplicity, this paper employs the term black-box to refer to the existing data-box techniques that are applied in a black-box manner.

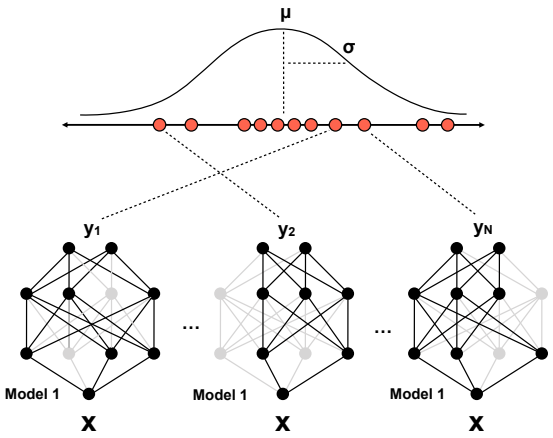


Fig. 2: Distribution approximated through MC-Dropout.

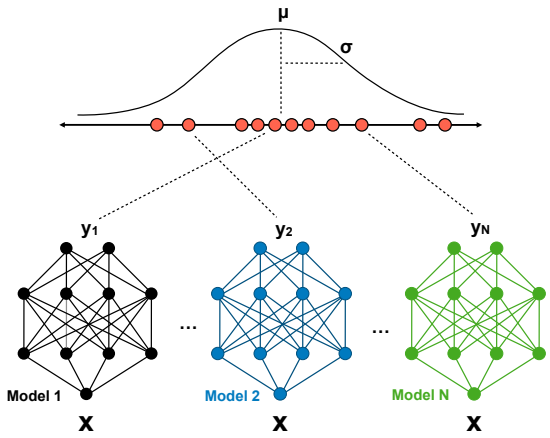


Fig. 3: Distribution approximated through Deep Ensembles.

MCD can be enabled based on the principle of Markov Chain Monte Carlo. When estimating predictive uncertainty with MC-Dropout, the dropout layers of the DNN are enabled also at inference time. Hence, predictions are no longer deterministic, being dependent on which nodes/links are randomly chosen by the network (see Figure 2). Therefore, given the same test data point ($X$ in the figure), the model will predict slightly different values every time the point is processed by the DNN, by "dropping" a selection of neurons across layers, except for the output layer.

This method can be regarded as an approximate Bayesian Neural Network (BNN) approach to uncertainty modeling. The Bayesian approach defines the model's likelihood, where Gaussian likelihood is often assumed for regression, with $\omega$ being the model parameters, $x$ the input and $y$ the output [35]:

$$p(y|x,\omega) = \mathcal{N}(avg(f_\omega(x)), var(f_\omega(x)))$$

MCD is used to generate samples interpreted as a probability distribution through Bayesian interpretation [34]: the value predicted by the DNN will be the mean ($avg$, or $\mu$ in Figure 2) of such probability distribution. Moreover, by collecting multiple predictions for input, each with a different realization of weights due to dropout layers, it is possible to account for model uncertainty as the variance ($var$, or $\sigma$ in Figure 2) of the observed probability distribution.

The rationale for using MC-Dropout is that supported inputs are expected to be characterized by low DNN uncertainties, whereas unsupported inputs are expected to increase it [36]. While being simple to implement, MC-Dropout is an intrusive approach, as it requires access to the existing DNN architectures, for which dropout layers need to be enabled also at testing time, or added if not already present [11].

Two hyperparameters influence the behaviour of MCD: (i) the number of stochastic forward passes and (ii) the dropout rate. While empirical guidelines exist [34], in this paper we aim to assess the effectiveness of MCD as a failure predictor for ADS testing under a large combination of these parameters.

### B. Deep Ensembles

The second considered UQ method involves another Bayesian method called Deep Ensembles [37] (DE). DE requires training multiple instances of the same model architecture on the same dataset while varying other factors to introduce randomicity. The ensemble predictions constitute an output distribution in which the variance of the ensemble characterizes the uncertainty (i.e., a larger variance implies larger uncertainty). Among the strategies to build DE, we recall bootstrapping, using different DNN architectures in terms of a number of layers and type of activation functions, random initialization of parameters along with a random shuffle of the datasets, and hyper-ensembles, in which ensembles with different hyperparameters are combined [11].

In this paper, we rely on random initialization of deep ensembles, which has shown promising results for many practical problems [11], [38]. Figure 3 provides a visual representation of this method. DE is a mixture model:

$$p(y|x) = \frac{1}{N}\sum_{n=1}^{N} p_n^*(y|x,\omega_n)$$

where the predictions are combined into one output $\mu$ (interpreted as a mixture of Gaussian distributions) and the variance of the outputs ($\sigma$ in Figure 2) measures the uncertainty.

Deep Ensembles provide a robust measure of uncertainty that is able to account for multiple sources of model and data uncertainty [11]. For DE, the main hyperparameter is the number of models ($N$). For large values of $N$, DE provides a precise implementation of the BNN approach, a theoretically grounded approach that provides the best uncertainty quantification while, however, being associated with a high computational cost. Thus, the trade-off between the precision of the BNN approximation and computational cost must be assessed in each application domain, such as ADS. An advantage of DE is that it is widely applicable, as it does not require any modification of any existing DNN. However, the computational overhead associated with training multiple models and loading them simultaneously in memory

during inference might be unacceptable for a large number of models. In this paper, we aim to assess the effectiveness and performance of DE as a failure predictor for ADS testing under a large number of ensemble sizes.

### C. Implementation

We implemented our codebase in Python and made it publicly available [25]. We support ADS models written in Tensorflow/Keras integrated in the Udacity simulator for self-driving cars [24]. Both UQ methods (MC-Dropout and Deep Ensembles) were tested on instances of NVIDIA's DAVE-2 [26] models. For MC-Dropout, a dropout layer was added between each layer of the original model.

## IV. Empirical Evaluation

### A. Research Questions

We consider the following research questions:

**$RQ_1$ (effectiveness):** How effective is UQ at predicting failures of ADS? What is the best configuration in terms of dropout rate and number of samples (for MCD) or number of models (for DE)? How does the effectiveness vary when considering different confidence levels?

**$RQ_2$ (prediction over time):** How does the prediction power of UQ change when considering different detection periods?

**$RQ_3$ (comparison):** How does UQ compare with SelfOracle [18] and ThirdEye [23] in terms of effectiveness?

**$RQ_4$ (performance):** What is the performance of running UQ in terms of time overhead in making predictions? How do the UQ methods compare with SelfOracle and ThirdEye?

The first research question ($RQ_1$) aims to assess whether our approach is able to attain a high failure prediction rate and which method (i.e., MC-Dropout, Deep Ensembles, and their parameters) yields the best prediction score. Failure prediction is only useful if it helps to anticipate a failure, which is studied in the second research question ($RQ_2$). To assess the usefulness of UQ methods over existing solutions, the third research question ($RQ_3$) compares UQ with two state-of-the-art failure predictors for ADS [18], [23]. The last research question ($RQ_4$) evaluates the runtime cost of each technique, to assess efficiency in conjunction with effectiveness.

### B. Experimental Setup

In this paper, we follow the same experimental setting of the original papers we compare against [18], [23], in terms of simulation platform, objects of study, and metrics. We briefly summarize the experimental setup next.

*1) ADS Under Test:* To implement DNN-based ADS, we use NVIDIA's DAVE-2 model [26], a reference model widely used as the object of study in prior related work [16], [18], [21], [39]–[43]. 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 a dropout rate of $0.05$ and ReLu activation function. For the experiments with SelfOracle and ThirdEye, we obtained the trained DAVE-2 models from the replication package of our baselines [18], [23], to make sure to test the same ADS used in the previous work. For UQ, we had no choice but to retrain DAVE-2 (details available in Section IV-B6).

*2) Driving Simulator:* We tested UQ through simulation-based testing, which is the standard practice for testing ADS and their behaviour prior to real-world deployment [44]–[47]. 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 [3], [4], [48], [49]. While our approach is independent from the chosen simulation platform, in our study to test the lane-keeping ADS we used the Udacity simulator for self-driving cars [24], a cross-platform driving simulator developed with Unity3D [50], used in the ADS testing literature [18], [19], [21], [41], [51], including our baselines [18], [23]. The simulator supports various closed-loop tracks for testing behavioural cloning ADS models, as well as the ability to generate changeable environmental perturbations (e.g., weather effects), which is useful to test an ADS on both nominal and unseen conditions. We chose the default sunny weather condition as the reference nominal scenario.

*3) Benchmark:* Concerning our evaluation set, we consider three existing datasets of simulations from previous work [23]. The first two datasets deal with failures induced by *out-of-distribution conditions* (OOD). An ADS that has been trained on some given nominal conditions and environment can fail in different instances of that environment. The first OOD benchmark ($\text{OOD}_{\text{extreme}}$) is characterized by severe illumination/weather conditions with respect to the nominal sunny scenario (see Figure 1). These conditions are available from the replication package of the SelfOracle paper [18] and account for 7 simulations with different degrees of extreme OOD conditions: day/night, rain, snow, fog, day/night + rain, day/night + snow, day/night + fog. The second OOD benchmark ($\text{OOD}_{\text{moderate}}$) consists of milder weather conditions without the strong luminosity changes present in the $\text{OOD}_{\text{extreme}}$ benchmark. Overall, concerning the OOD benchmarks, 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). The third benchmark (Mutants) consists of faulty ADS models produced by mutation testing [42]. In this case, the ADS drives under nominal (sunny) conditions, but it can occasionally fail due to inadequate training, a frequent scenario during the development process of an ADS model (i.e., data collection, training, and testing is an iterative process [21]). Overall, the evaluation set comprises 265 failures that our approach is expected to detect timely. Both scenarios are of interest to our work, as a failure predictor should be agnostic about the conditions that cause the failures (i.e., unknown inputs or DNN model bugs). Moreover, to estimate the threshold used by UQ methods, the evaluation set includes 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.

*4) Detection Windows in Evaluation Set:* The Udacity simulator automatically labels individual failing frames as either nominal or failing, according to whether the ADS was

on track or off-track, respectively. We focus on the part of the simulation *preceding each failure*, whereas the frames labeled as failing are not considered. When a simulation exhibits multiple failures, we assess each failure individually. Differently from the compared papers [18], [23], for all benchmarks, we calculate the actual frame rate of each simulation, instead of using a fixed window size of 15 frames. This choice was motivated by the fact the three benchmarks were captured on different machines and hardware, at different frame rates. Consequently, using a fixed window size would fail to uniformly represent simulation time across all datasets, making it challenging to fairly evaluate the performance of our predictors.

*5) Baselines:* As described in Section II-C, we use two baselines for UQ. Concerning SelfOracle we consider the best configuration presented in the original paper, which uses a variational autoencoder [52] (VAE) with a latent size of 2, trained to minimize the MSE (see Section II-C) between the original and reconstructed nominal images (sunny). Regarding ThirdEye, we assessed the best configuration that includes heatmap derivative as a summarization method.

*6) Configurations:* For both UQ methods, we trained lane-stable DAVE-2 models using an existing dataset [18] with more than $32k$ images on nominal sunny conditions following two different track orientations (normal, reverse), and additional data for recovery. Each image is labeled with the human expert-provided ground truth steering angle value for that driving image. The maximum driving speed of the driving model was 30 mph during data generation, the default value in the Udacity simulator.

For MC-Dropout, we trained several DAVE-2 models varying two parameters. The first parameter is the dropout rate, which we vary in the range $[0.05, 0.1, 0.15, 0.20, 0.25, 0.30, 0.35]$. Models with a dropout rate higher than $0.40$ were disregarded for not being able to complete a lap in the simulator. The second parameter is the number of samples, which we vary in the range $[2, 3, 4, 5, 10, 20, 32, 64, 128]$. For Deep Ensembles, we trained several DAVE-2 models varying the number of models in the ensemble, considering the range $[2, 3, 4, 5, 6, 7, 10, 30, 50, 70, 90, 100, 120]$.

The number of epochs was set to 50, with a batch size of 128 and a learning rate of 0.0001. We used early stopping with a patience of 10 and a minimum loss change of 0.0005 on the validation set. The network uses the Adam optimizer [53] to minimize the MSE between the predicted steering angles and the ground truth value. We used data augmentation to mitigate the lack of image diversity in the training data. Specifically, 60% of the data was augmented through different image transformation techniques (e.g., flipping, translation, shadowing, brightness). We cropped the images to 80x160 and converted them from RGB to YUV color space. We only retained solid models for testing, i.e., models able to drive multiple laps in each track under nominal conditions without showing any misbehavior in terms of crashes or out-of-track events. This should also provide more guarantees about the quality of the uncertainty score estimations obtained from white box access to the models.

Overall, our experiment includes 232 models under test. For MC-Dropout, we trained 63 final models (7 dropout rates $\times$ 9 number of samples) for parameter optimization and did further testing on the best dropout rates to study the distribution. For Deep Ensembles, we trained 138 different models and built 30 different ensembles. For smaller-sized ensembles [2-5] we tested various combinations of models to study their effectiveness. As our evaluation set comprises 380,717 images, overall we computed 15,723,347 uncertainty scores in our experiments (11.5 days computing time).

*7) Metrics used for Analysis:* To answer $RQ_1$, $RQ_2$, and $RQ_3$, we apply a window function on non-overlapping, fixed length, sequences of scores, returning the *maximum* score within a window. In previous work [23], the *arithmetic mean* of the scores within a window was also used, with less promising results. Therefore, in this paper, we limit our investigation to the maximum window function. The sets of (windowed) uncertainty confidence scores represent a model of normality collected in nominal driving conditions using different methods for computing the uncertainty profiles. Following existing literature [18], [23], we use probability distribution fitting to obtain a statistical model of the uncertainty scores. We set a threshold $\gamma$ for the expected false alarm rate in nominal conditions and estimate the shape $\kappa$ and scale $\theta$ of a fitted Gamma distribution of the uncertainty scores to ensure the expected false alarm rate is below the chosen threshold $\gamma$ [18]. In this study, we experiment with different thresholds, varying $\gamma$ in the range $[0.95, 0.99, 0.999, 0.9999, 0.99999]$, hence expanding substantially the $\gamma$ threshold ranges considered previously (ThirdEye was only evaluated for $\gamma = 0.95$, whereas SelfOracle was evaluated for $\gamma = 0.95$ and $\gamma = 0.99$).

We compute the true positives as the number of correct failure predictions within a detection window and the false negatives as the number of missed failure predictions when our framework does not trigger an alarm in a detection window. The false positives and true negatives are measured using nominal simulations to which analogous windowing is applied. Our primary goal is to achieve a 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 reported by a technique. Consistent with previous work [23], we consider the $F_{beta}$ score [54], with $\beta = 3.0$, as a weighted balance between precision and recall ($F_3 = \frac{10 \cdot \text{Precision} \times \text{Recall}}{9 \cdot \text{Precision} + \text{Recall}}$), staying consistent with previous work. 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 [54] as it means a missed failure detection. In contrast, in our setting, the cost associated with false positives (false alarms) is relatively lower compared to false negatives.

We also compute the threshold-independent metric AUC-ROC (area under the curve of the Receiver Operating Char-

TABLE I: $RQ_{1\text{-}2\text{-}3}$: Results for the best failure predictors. Bold = average $F_3$ scores; grey = best $F_3$ scores.

| | | MCD5 S32 conf = 0.99 | | | MCD5 S64 conf = 0.99 | | | MCD5 S128 conf = 0.99 | | | DE5 conf = 0.999 | | | DE10 conf = 0.999 | | | DE50 conf = 0.999 | | | SelfOracle conf = 0.99 | | | ThirdEye conf = 0.95 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 | $F_3$ |
| **$OOD_{extreme}$** | 1 | 22 | 93 | 69 | 19 | 100 | 69 | 22 | 93 | 69 | 42 | 100 | 87 | 42 | 100 | 87 | 100 | 100 | 100 | 73 | 100 | 96 | 19 | 93 | 65 |
| | 2 | 23 | 100 | 73 | 19 | 95 | 66 | 20 | 88 | 65 | 42 | 100 | 87 | 42 | 100 | 87 | 100 | 100 | 100 | 73 | 96 | 93 | 19 | 95 | 66 |
| | 3 | 23 | 96 | 71 | 17 | 82 | 59 | 22 | 89 | 67 | 43 | 100 | 87 | 43 | 100 | 87 | 100 | 100 | 100 | 70 | 89 | 86 | 19 | 93 | 66 |
| | avg | 22 | 96 | **71** | 18 | 92 | **65** | 21 | 90 | **67** | 42 | 100 | **87** | 42 | 100 | **87** | 100 | 100 | **100** | 72 | 95 | **92** | 19 | 94 | **66** |
| **$OOD_{moderate}$** | 1 | 31 | 100 | 80 | 30 | 98 | 79 | 30 | 98 | 79 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 51 | 98 | 89 | 13 | 87 | 54 |
| | 2 | 27 | 86 | 69 | 26 | 83 | 67 | 25 | 81 | 65 | 100 | 97 | 97 | 100 | 98 | 98 | 100 | 100 | 100 | 47 | 91 | 83 | 11 | 75 | 47 |
| | 3 | 21 | 63 | 51 | 21 | 63 | 51 | 20 | 63 | 51 | 72 | 70 | 70 | 89 | 79 | 79 | 72 | 70 | 70 | 33 | 62 | 57 | 10 | 62 | 40 |
| | avg | 26 | 83 | **67** | 26 | 81 | **66** | 25 | 81 | **65** | 91 | 89 | **89** | 96 | 92 | **93** | 91 | 90 | **90** | 44 | 84 | **76** | 12 | 75 | **47** |
| **Mutants** | 1 | 65 | 100 | 94 | 65 | 99 | 94 | 65 | 100 | 94 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 77 | 82 | 81 | 44 | 99 | 87 |
| | 2 | 65 | 98 | 93 | 64 | 97 | 92 | 64 | 97 | 92 | 100 | 96 | 96 | 100 | 97 | 97 | 100 | 97 | 97 | 61 | 49 | 50 | 44 | 97 | 86 |
| | 3 | 60 | 88 | 84 | 59 | 85 | 81 | 59 | 87 | 83 | 100 | 81 | 82 | 100 | 87 | 87 | 94 | 81 | 82 | 56 | 41 | 41 | 41 | 91 | 80 |
| | avg | 63 | 95 | **90** | 63 | 94 | **89** | 63 | 95 | **90** | 100 | 92 | **93** | 100 | 95 | **95** | 98 | 93 | **93** | 65 | 57 | **57** | 43 | 95 | **84** |
| *Average (All)* | 1 | 39 | 98 | 81 | 38 | 99 | 81 | 39 | 97 | 81 | 81 | 100 | 96 | 81 | 100 | 96 | 100 | 100 | 100 | 67 | 94 | 89 | 25 | 93 | 69 |
| | 2 | 38 | 95 | 79 | 36 | 92 | 75 | 37 | 89 | 74 | 81 | 97 | 93 | 81 | 98 | 94 | 100 | 99 | 99 | 60 | 79 | 75 | 25 | 89 | 66 |
| | 3 | 34 | 83 | 69 | 32 | 77 | 64 | 34 | 80 | 67 | 71 | 84 | 80 | 77 | 88 | 85 | 89 | 84 | 84 | 53 | 64 | 61 | 24 | 82 | 62 |
| | avg | 37 | 92 | **76** | 36 | 89 | **73** | 36 | 88 | **74** | 78 | 94 | **90** | 79 | 96 | **92** | 96 | 94 | **94** | 60 | 79 | **75** | 24 | 88 | **66** |

acteristics), which we use to choose the top three models as presenting the results for all models would be infeasible. For $RQ_2$, for each failure, we adopt a detection window granularity equal to one second of simulation in the Udacity simulator and we consider window sizes from 1 to 3 seconds prior to the failures (time to failure, TTF for short). Previous studies in the Udacity simulator [36] indicate a TTF of 3 seconds as sufficient to avoid failures at 30 mph, which is the constant cruising speed of the ADS in the simulator.

To answer $RQ_4$, we compute the execution time (in milliseconds *ms*) and RAM usage during inference using the Python tool `mprofile` [55] on a machine featuring an AMD Ryzen 7 3800XT 8-Core (16 Threads) Processor, 32GB system RAM and a NVIDIA 3070 GPU with 8GB of VRAM. All models were evaluated using two laps under normal conditions for a total evaluation set consisting of 11,031 images. All inferences were computed using the CPU only with all 16 (virtual) cores enabled. For Deep Ensembles, all models of an ensemble were loaded into memory and performed the inference concurrently. For MC-Dropout, the model was loaded into memory and performed the inference concurrently, running the inference process multiple times as multiple parallel threads. For SelfOracle and ThirdEye, the cache was cleared, forcing the models to compute the heatmaps during inference instead of relying on pre-computed values.

### C. Results

*1) Effectiveness ($RQ_1$):* For MCD, the top three configurations from our experiments are MCD models with dropout rate=0.05 and number of samples 32, 64, and 128. In the rest of the paper, we refer to them as MCD-5 S32, MCD-5 S64, and MCD-5 S128, respectively. For DE, the top three configurations from our experiments are DE with 5, 10, and 128 models, referred to as DE5, DE10, and DE50 next. Figure 4 reports the three models for each UQ method and the two baselines at

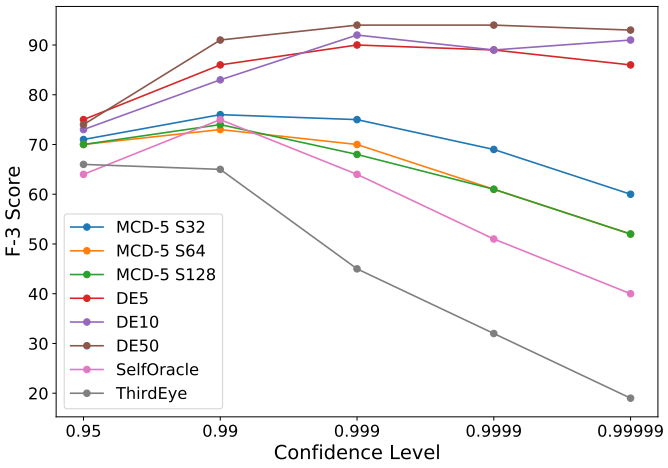


Fig. 4: $RQ_1$: $F_3$ scores for the best failure predictors across all confidence levels.

different confidence values. Deep Ensembles models perform well across all confidence levels. MC-Dropout models perform well at confidence levels $\gamma = 0.95$ and $\gamma = 0.99$ and worse with higher confidence levels. Consequently, in the rest of the paper, we report detailed results considering the optimal confidence threshold for each model.

Table I presents the effectiveness results for the top three configurations of UQ (MC-Dropout, Deep Ensembles), SelfOracle, and ThirdEye. Results are averaged across conditions, split between external unknown conditions ($OOD_{extreme}$ and $OOD_{moderate}$) and internal uncertain conditions (Mutants). For each condition, we evaluate failure detection with a detection window of 1-3 seconds and also report the average of these scores. The effectiveness metrics consider the optimal confidence threshold for each model (Figure 4). Precision (Pr) is

measured in anomalous conditions, which explains why it is lower than the expected value associated with the confidence threshold in most cases. Due to space constraints, in this section, we only comment on the average $F_3$ scores over all benchmarks. On average, UQ with MC-Dropout reaches a $F_3$ score of 73−76c. UQ with Deep Ensembles, on the other hand, performs better with $F_3$ scores of $90 - 94$. For the MCD-5 model, increasing the sample size does not improve the effectiveness but rather causes a slight drop in the $F_3$ score. The precision for MCD-5 remains relatively low across all sample sizes larger than 32, indicating that the false positive rate does not improve with a higher number of samples. For Deep Ensembles, the theoretical best performance is DE50 (i.e., an ensemble of 50 models) with an $F_3$ of 94, outperforming any other configuration. In practice, though, a DE of 50 models might be computationally expensive, therefore DE5 or DE10 are more likely to be used. All Deep Ensembles models have a high recall and a low false positive rate (i.e., high precision).

> **RQ$_1$**: *UQ with Deep Ensembles (5/50 models) is the best-performing failure predictor for ADS, achieving the highest failure prediction rates across all conditions ($F_3$ = 90-94%).*

*2) Prediction Over Time (RQ$_2$):* Table I reports the effectiveness considering different time to failure (TTF, Column 2). In principle, failure prediction should get more challenging as we move farther from the failure. This is confirmed for all configurations of UQ (considering the average scores) with the prediction power dropping ($F_3$) slightly when we move from a 1-second detection window to a 2-second window and a larger drop when considering a 3-second window. The best MC-Dropout model performance drops by -3.5% and -14.8% at 2 and 3 seconds TTF respectively, compared to 1 second TTF. The best Deep Ensembles model performance drops by -1% and -16% respectively. When we look at the $\text{OOD}_{\text{extreme}}$ benchmark, we observe that Deep Ensembles of all sizes do not drop any predictive power up to 3 seconds TTF, with DE50 predicting all failures.

> **RQ$_2$**: *On average, the effectiveness of the best configurations of UQ drops by 16% average $F_3$ up to 3 seconds before the failures. The effectiveness of UQ with Deep Ensembles remains high under $\text{OOD}_{extreme}$ conditions (no decrease in $F_3$) up to 3 seconds before the failure.*

*3) Comparison (RQ$_3$):* Considering the average $F_3$ scores across benchmarks from Table I, the best configurations of both UQ methods are superior to SelfOracle and ThirdEye at predicting misbehaviours. MC-Dropout with a 5% dropout rate and 32 sample size is comparable to SelfOracle with a +1% improvement in $F_3$ score. Compared to ThirdEye, MCD5-S32 is +15% better at predicting misbehaviour ($F_3$). Deep Ensembles 50 outperforms both SelfOracle and ThirdEye, with an improvement of +25% and +42% in average $F_3$ scores, respectively. On the $\text{OOD}_{\text{extreme}}$ benchmark, UQ scores a +8.7% and +51% increase in $F_3$ w.r.t. SelfOracle and ThirdEye. For $\text{OOD}_{\text{moderate}}$ conditions, average $F_3$ scores raise up to 93, for DE10, whereas the best $F_3$ from our baseline (SelfOracle) is 76%. For Mutants, our results show a remarkable difference in effectiveness between UQ over SelfOracle and ThirdEye. Both MC-Dropout and Deep Ensembles score higher with average $F_3$ scores in the range of 89-95, a +66.7% w.r.t. SelfOracle in $F_3$ and 13% w.r.t. ThirdEye.

Overall, average results for $F_3$ show significant improvements of UQ over previous experiments. For Deep Ensembles, this finding holds independent of the configuration being used and the reaction period considered.

We assessed the statistical significance of these differences using the non-parametric Mann-Whitney U test [56] (with $\alpha = 0.05$) and the magnitude of the differences using the Cohen's $d$ effect size [57]. The difference in $F_3$ score between Deep Ensembles and SelfOracle and ThirdEye were found to be statistically significant ($p$-value $< 0.05$) with medium and large effect sizes. As expected by looking at the average $F_3$ scores of Table I, there is no statistically significant difference between MCD and SelfOracle ($p$-value $\geq 0.05$), whereas the difference with ThirdEye is statistically significant with medium effect size.

> **RQ$_3$**: *UQ with Deep Ensembles outperforms SelfOracle and ThirdEye in terms of failure prediction under all conditions, with statistical significance.*

*4) Performance (RQ$_4$):* Figure 5 shows the results of the different models in ms per iteration/image. Deep Ensembles performed best with the DE5 employing 2.5 ms/image and the DE50 employing 17.7 ms/image. SelfOracle has a similar performance to larger deep ensembles with 12.8 ms/image.

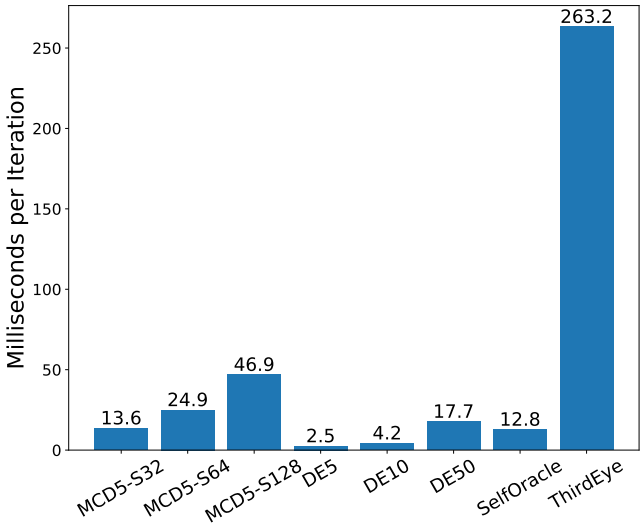


Fig. 5: RQ$_4$: Computational overhead (ms/iteration).

MC-Dropout performs worse than deep ensembles and SelfOracle. MCD-5 S32 took 13.6 ms/image and 46.9 ms/image for MCD-5 S128. Both UQ methods, as expected, decrease in performance when either the number of models or the sample size increases. ThirdEye, as seen in Figure 5, takes significantly longer than any other method to process an image, being 105× *slower* than Deep Ensembles. This performance is expected, as ThirdEye needs to compute the heatmap for each image, which is a computationally expensive process.

Concerning the memory usage of the different models, we do not report extensive results, but we discuss a few interesting insights. MC-Dropout used the least amount of memory considering its best configuration (635 MB). Deep Ensembles 50 used a larger amount of memory, requiring loading all models into memory simultaneously (1.37 GB). The size of each model itself (used in MC-Dropout or the Deep Ensembles) is approximately 4.7 MB. SelfOracle and ThirdEye used the most amount of memory, requiring 27.6 GB and 7.3 GB of memory respectively.

> **RQ$_4$**: *Small Deep Ensembles are the most computationally efficient outperforming ThirdEye and SelfOracle. Particularly, DE5 and DE10 employ on average less than 5 ms/image.*

### D. Threats to Validity

*1) Internal validity:* All variants of UQ, SelfOracle, and ThirdEye were compared under identical experimental settings and on the same evaluation set. Thus, 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 [18] and ThirdEye [23] papers. Regarding the simulation platform, to allow a fair comparison, we used the Udacity simulator adopted in analogous failure prediction studies [18], [23]. However, it is important to note our approach is independent of the chosen simulation platform. Other open-source propositions are available, such CARLA [58], LGSVL [59], and BeamNG [60]. CARLA and LGSVL mostly deal with urban environments with static and dynamic obstacles, whereas BeamNG is conceptually similar to Udacity as it was used in similar lane-keeping testing studies [5], [9], [61]. We discard commercial close-source solutions such as Siemens PreScan [62], ESI Pro-SiVIC [63], and PTV VISSIM [64] as they do not allow full replicability of our results and also focus on urban scenarios or other ADS tasks such as automated valet parking or breaking assistance.

*2) External validity:* The limited number of self-driving systems in our evaluation constitutes a threat to the generalizability of our results to other ADS. Moreover, results may not generalize, or generalize differently, when considering other simulation platforms than Udacity. For the uncertainty scores, we considered two quantification methods, and the effectiveness of our tool may change when considering different strategies. To mitigate this issues, we selected the most popular techniques for computing uncertainties in regression deep neural networks, as outlined in Weiss and Tonella [12].

*3) Reproducibility:* All our results, the source code, and the simulator are accessible and can be reproduced [25].

## V. Discussion

### A. UQ for Failure Prediction

Our research emphasizes the intricate nature and diverse range of failure scenarios that runtime monitoring techniques must address. Uncertainty scores, usually employed quantitatively by humans to understand deep neural network mispredictions, were used in this study as a cumulative error scoring function over time. This approach assumes that these scores contain valuable information for assessing the behavior of DNNs and, by extension, of the autonomous driving systems that rely on them.

Our approach relies on the efficacy of uncertainty scores as a technique for assessing the nominal driving behavior of ADS. A well-trained DNN would excel in capturing relevant structures in an image, such as road lanes, resulting in more precise uncertainty scores compared to inadequately trained DNNs. Furthermore, methods for quantifying uncertainty provide a more transparent and efficient means of evaluating ADS behavior than opaque data- or black-box techniques. Our findings confirm that UQ methods outperform existing techniques in both out-of-distribution and mutation testing scenarios.

### B. Discussing UQ Configurations

In our benchmarks, UQ using MC-Dropout exhibited superior performance on the Mutants dataset compared to both $\text{OOD}_{\text{moderate}}$ and $\text{OOD}_{\text{extreme}}$. It demonstrated the capability to predict 95% failures with an acceptable precision, up to 3 seconds in advance. This observation underscores the effectiveness of MC-Dropout as a reliable metric for understanding internal model uncertainty. Conversely, UQ with Deep Ensembles consistently delivered remarkable prediction results across all benchmarks. Even for different confidence levels (Figure 4), Deep Ensembles consistently outperformed alternative methods. Our findings confirm that Deep Ensembles excel at capturing uncertainty from diverse sources and outshines MC-Dropout [11].

While the theoretical best ensemble with 50 models may not be practical for real-world applications in ADS, our Deep Ensembles with only 5 models outperformed all other techniques and exhibited robust uncertainty estimates, with performance similar to DE50. Taking into account computational runtime, we found that smaller Deep Ensembles were more efficient than MC-Dropout. This advantage stems from the ability to load and run multiple models concurrently, provided the hardware can support the model sizes. In contrast, MC-Dropout requires less memory but still needs to run the inference multiple times (32-128), making it less competitive than DE5 or DE10. Furthermore, implementing MC-Dropout necessitates modifications to the ADS model. Considering all

these factors, our comprehensive evaluation identifies UQ with Deep Ensembles as the optimal configuration, delivering the best results in our study.

### C. Comparison with Other Approaches

As a baseline for our experiment, we used SelfOracle and ThirdEye from previous literature. In contrast to the previous experiment, we modified the evaluation as described in Section IV-B4 by implementing a dynamic window calculation for the $OOD_{extreme}$ benchmark. This allowed us to compare the benchmark scores more objectively. However, this caused the magnitude of the results for SelfOracle and ThirdEye to change [18], [23]. UQ with Deep Ensembles is a clear improvement over the baselines in terms of effectiveness and computation time. While each of the two baselines performed well in a specific benchmark (SelfOracle in $OOD_{extreme}$ and ThirdEye in Mutants), UQ with Deep Ensembles performs well across all benchmarks. Notably, even hybrid approaches with MCD + SelfOracle or MCD + ThirdEye are not expected to achieve higher scores than DE as they require more computational resources than Deep Ensembles.

## VI. Related Work

### A. Anomaly Detection in Autonomous Driving

We already discussed SelfOracle [18] and ThirdEye [23], for which we performed an explicit empirical comparison in this work. Similarly to SelfOracle, DeepGuard [20] uses the reconstruction error by VAEs to prevent collisions of vehicles with the roadside. DeepRoad [16] uses embeddings created from features extracted by VGGNet [65] to validate driving images based on the distance to the training set. In other works [36], [51], continual learning is used to minimize the false positives of a black-box failure predictor. Hell et al. [19] evaluate VAEs, Likelihood Regret, and the generative modelling SSD, for ADS testing on OOD detection in the CARLA simulator. Michelmore et al. [29], [30] use Bayesian inference methods for probabilistic safety estimation. Henriksson et al. [13] use the negative of the log-likelihood as a black-box anomaly score. Borg et al. [66] propose to pair OOD detection with VAEs with object detection for an automated emergency braking system. Strickland et al. [67] use an LSTM solution with multiple metrics to predict collisions with vehicles at crossroads. Ayerdi et al. [68] propose the use of metamorphic oracles to supervise a DNN-based ADS.

Our approach reports extensive simulation-based testing results for both the effectiveness and efficiency of uncertainty quantification methods. For a broad overview of anomaly detection techniques in autonomous driving, we refer the reader to the survey by Bogdoll et. al [69].

### B. Uses of Uncertainty in Software Engineering

Uncertainty quantification is also popular in software engineering, especially in the context of cyber-physical systems. Hu et al. [70] used uncertainty quantification to improve the performance of transfer learning for evolving digital twins of industrial elevators. Similarly, the PPT method [71] proposes uncertainty-aware transfer learning for digital twins. PPT is evaluated on cyber-physical systems and ADS, with positive results in terms of the effectiveness of uncertainty quantification for reducing the Huber loss in both case studies.

Weiss et al. [12] report an empirical study of uncertainty quantification methods that are used to implement supervisors for DNNs. The evaluation is done at the model-level, for four classification classification datasets. Results show that the uncertainty monitors were able to increase the accuracy of the DNNs when supervised. Differently, in this paper, we use uncertainty quantification to inform a system-level failure predictor for ADS.

### C. Generic OOD Detectors

Generic detectors of out-of-distribution samples have been proposed, which we describe for completeness. AutoTrainer [72] analyzes the training process of a DNN to automatically repair when metrics such as accuracy used during training degrade. In contrast, UQ operates at testing time, not at training time, to recognize uncertain execution conditions of an ADS, whereas AutoTrainer operates at training time.

Zhang et al. [73] propose an algorithm for the automatic detection of OOD inputs based on the notion of relative activation and deactivation states of a DNN. 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 [15] helps answer these questions, but the evaluation of the DNN prediction is performed for individual samples. UQ works with normal feed-forward passes, making them computationally more efficient and easier to integrate into the ADS development process.

## VII. Conclusions and Future Work

In this paper, we describe and evaluate white-box failure predictors based on uncertainty quantification methods. We use them to estimate the confidence of a DNN-based ADS in response to unseen execution contexts. Our results show that UQ methods can anticipate many potentially safety-critical failures by several seconds, with a low or zero false alarm rate in anomalous conditions, and a fixed negligible expected false alarm rate in nominal conditions, outperforming two existing solutions from the literature.

Future work includes extending the comparison to other benchmarks, multi-module ADS, simulators, and ADS case studies such as urban driving for which revisions of the existing methods would be necessary, or alternative confidence score synthesis methods. Furthermore, we intend to broaden our scope by enhancing the detection of more subtle forms of driving quality degradation, such as erratic driving behavior. Additionally, we will explore the implementation of self-healing mechanisms within the simulator and extend our evaluation on physical driving testbeds.

## Acknowledgements

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

## REFERENCES


[1] 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.

[2] 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.

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

[4] V. G. Cerf, "A comprehensive self-driving car test," *Commun. ACM*, vol. 61, no. 2, pp. 7–7, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3177753

[5] A. Gambi, M. Mueller, and G. Fraser, "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*, ser. ISSTA 2019. New York, NY, USA: ACM, 2019, pp. 318–328. [Online]. Available: http://doi.acm.org/10.1145/3293882.3330566

[6] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, May 2018, pp. 1016–1026.

[7] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 143–154. [Online]. Available: http://doi.acm.org/10.1145/3238147.3238192

[8] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2016, pp. 63–74.

[9] V. Riccio and P. Tonella, "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*, ser. ESEC/FSE '20, 2020.

[10] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "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., 2015.

[11] W. He and Z. Jiang, "A survey on uncertainty quantification methods for deep neural networks: An uncertainty source perspective," 2023.

[12] M. Weiss and P. Tonella, "Fail-safe execution of deep learning based systems through uncertainty monitoring," in *IEEE 14th International Conference on Software Testing, Validation and Verification*, ser. ICST '21. IEEE, 2021.

[13] J. Henriksson, C. Berger, M. Borg, L. Tornberg, C. Englund, S. R. Sathyamoorthy, and S. Ursing, "Towards structured evaluation of deep neural network supervisors," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, Apr. 2019.

[14] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 1039–1049. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00108

[15] Y. Xiao, I. Beschastnikh, D. S. Rosenblum, C. Sun, S. Elbaum, Y. Lin, and J. S. Dong, "Self-checking deep neural networks in deployment," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 372–384.

[16] 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 International Conference on Automated Software Engineering*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 132–142. [Online]. Available: http://doi.acm.org/10.1145/3238147.3238187

[17] H. Wang, J. Xu, C. Xu, X. Ma, and J. Lu, "Dissector: Input validation for deep learning applications by crossing-layer dissection," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2020, pp. 727–738.

[18] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of 42nd International Conference on Software Engineering*, ser. ICSE '20. ACM, 2020, p. 12 pages.

[19] F. Hell, G. Hinz, F. Liu, S. Goyal, K. Pei, T. Lytvynenko, A. Knoll, and C. Yiqiang, "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*, ser. CSCS '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3488904.3493382

[20] M. Hussain, N. Ali, and J.-E. Hong, "Deepguard: A framework for safeguarding autonomous driving systems from inconsistent behaviour," *Automated Software Engg.*, vol. 29, no. 1, may 2022. [Online]. Available: https://doi.org/10.1007/s10515-021-00310-0

[21] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing Machine Learning based Systems: A Systematic Mapping," *Empirical Software Engineering*, 2020.

[22] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," in *Proceedings of 42nd International Conference on Software Engineering*, ser. ICSE '20. ACM, 2020, p. 12 pages.

[23] A. Stocco, P. J. Nunes, M. d'Amorim, and P. Tonella, "ThirdEye: Attention maps for safe autonomous driving systems," in *Proceedings of 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. IEEE/ACM, 2022.

[24] Udacity, "A self-driving car simulator built with Unity," https://github.com/udacity/self-driving-car-sim, 2017, online; accessed 25 October 2023.

[25] "Replication package." https://github.com/ast-fortiss-tum/misbehaviour-prediction-with-uncertainty-quantification, 2023.

[26] M. Bojarski, D. D. 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. [Online]. Available: http://arxiv.org/abs/1604.07316

[27] N. H. T. S. A. U.S. Department of Transportation, "Pre-crash scenario typology for crash avoidance research," 2007.

[28] T. R. I. . International Organization for Standardization, "Road vehicles - safety of the intended functionality," 2019.

[29] R. Michelmore, M. Kwiatkowska, and Y. Gal, "Evaluating uncertainty quantification in end-to-end autonomous driving control," *CoRR*, vol. abs/1811.06817, 2018.

[30] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, and M. Kwiatkowska, "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, 2020, pp. 7344–7350. [Online]. Available: https://doi.org/10.1109/ICRA40945.2020.9196844

[31] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, pp. 457–506, 2021.

[32] S. Bjarnadottir, Y. Li, and M. G. Stewart, "Climate adaptation for housing in hurricane regions," in *Climate Adaptation Engineering*. Elsevier, 2019, pp. 271–299.

[33] M. Weiss and P. Tonella, "Uncertainty-wizard: Fast and user-friendly neural network uncertainty quantification," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 436–441.

[34] Y. Gal and Z. Ghahramani, "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*, ser. ICML '16. JMLR.org, 2016.

[35] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information fusion*, vol. 76, pp. 243–297, 2021.

[36] A. Stocco and P. Tonella, "Confidence-driven weighted retraining for predicting safety-critical failures in autonomous driving systems," *Journal of Software: Evolution and Process*, 2021. [Online]. Available: https://doi.org/10.1002/smr.2386

[37] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[38] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv preprint arXiv:1912.02757*, 2019.

[39] 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: ACM, 2017, pp. 1–18. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132785

[40] 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: ACM, 2018, pp. 303–314. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180220

[41] 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.

[42] N. Humbatova, G. Jahangirova, and P. Tonella, "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*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 67–78. [Online]. Available: https://doi.org/10.1145/3460319.3464825

[43] 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.

[44] 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.

[45] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang, "Investigation into the state-of-the-practice autonomous driving testing," 2021. [Online]. Available: https://arxiv.org/abs/2106.12233

[46] A. Stocco, B. Pulfer, and P. Tonella, "Mind the Gap! A Study on the Transferability of Virtual vs Physical-world Testing of Autonomous Driving Systems," *IEEE Transactions on Software Engineering*, 2022.

[47] ——, "Model vs system level testing of autonomous driving systems: A replication and extension study," *Empirical Softw. Engg.*, vol. 28, no. 3, may 2023. [Online]. Available: https://doi.org/10.1007/s10664-023-10306-x

[48] BGR Media, LLC, "Waymo's self-driving cars hit 10 million miles," https://techcrunch.com/2018/10/10/waymos-self-driving-cars-hit-10-million-miles, 2018, online; accessed 25 October 2023.

[49] "Waymo Driver," https://waymo.com/waymo-driver/, 2021.

[50] "Unity3d." https://unity.com, 2021.

[51] A. Stocco and P. Tonella, "Towards anomaly detectors that learn continuously," in *Proceedings of 31st International Symposium on Software Reliability Engineering Workshops*, ser. ISSREW 2020. IEEE, 2020.

[52] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," 2015.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:6628106

[54] D. C. Blair, "Information retrieval, 2nd ed. c.j. van rijsbergen. london: Butterworths; 1979: 208 pp. price: $32.50," *Journal of the American Society for Information Science*, vol. 30, no. 6, pp. 374–375, 1979. [Online]. Available: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630300621

[55] T. Palpant, "mprofile," 1 2023. [Online]. Available: https://pypi.org/project/mprofile/

[56] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, p. 80, Dec. 1945. [Online]. Available: https://doi.org/10.2307/3001968

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

[58] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: an open urban driving simulator," *CoRR*, vol. abs/1711.03938, 2017. [Online]. Available: http://arxiv.org/abs/1711.03938

[59] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," *arXiv preprint arXiv:2005.03778*, 2020.

[60] BeamNG GmbH, "BeamNG.research," https://beamng.tech/, 2018, online; accessed 25 October 2023.

[61] M. Biagiola, A. Stocco, V. Riccio, and P. Tonella, "Two is better than one: Digital siblings to improve autonomous driving testing," 2023.

[62] S. D. I. Software, "Simcenter prescan," https://www.plm.automation.siemens.com/global/en/products/simcenter/prescan.html, 2023.

[63] E. Group, "Esi prosivic," https://myesi.esi-group.com/downloads/software-downloads/pro-sivic-2021.0, 2021.

[64] VISSIM, "VISSIM website," https://www.ptvgroup.com/en-us/products/ptv-vissim, 2023.

[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition."

[66] M. Borg, J. Henriksson, K. Socha, O. Lennartsson, E. S. Lönegren, T. Bui, P. Tomaszewski, S. R. Sathyamoorthy, S. Brink, and M. H. Moghadam, "Ergo, smirk is safe: A safety case for a machine learning component in a pedestrian automatic emergency brake system," 2022. [Online]. Available: https://arxiv.org/abs/2204.07874

[67] M. Strickland, G. Fainekos, and H. Ben Amor, "Deep predictive models for collision risk assessment in autonomous driving," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018*, ser. Proceedings - IEEE International Conference on Robotics and Automation. Institute of Electrical and Electronics Engineers Inc., 9 2018, pp. 4685–4692.

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

[69] D. Bogdoll, M. Nitsche, and J. M. Zöllner, "Anomaly detection in autonomous driving: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2204.07974

[70] Q. Xu, S. Ali, T. Yue, and M. Arratibel, "Uncertainty-aware transfer learning to evolve digital twins for industrial elevators," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 1257–1268.

[71] Q. Xu, T. Yue, S. Ali, and M. Arratibel, "Pretrain, prompt, and transfer: Evolving digital twins for time-to-event analysis in cyber-physical systems," *arXiv preprint arXiv:2310.00032*, 2023.

[72] X. Zhang, J. Zhai, S. Ma, and C. Shen, "Autotrainer: An automatic dnn training problem detection and repair system," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 359–371.

[73] Z. Zhang, P. Wu, Y. Chen, and J. Su, "Out-of-distribution detection through relative activation-deactivation abstractions," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 150–161.