Bootstrapping Aggregation of Neural Models for Prediction of Intelligent Vehicle Performance with Quantified Uncertainty: Safety Assessment, Failure Analysis, and Online Speed Recommendations

Antonello Cherubini, Gastone Pietro Rosati Papini, Alice Plebe, Mattia Piazza, Mauro Da Lio, *Member, IEEE*

Abstract—**Highly automated vehicles are complex systems, and ensuring their safe operation within their Operational Design Domain (ODD) presents significant challenges. Diagnosing failure modes and updating these systems are even more demanding tasks. This paper introduces a method to assist with the assessment, diagnosis, and updating of these systems through the development of a stochastic model that predicts safety outcomes (collision, near-miss, or safe state) with quantified uncertainty in any parametrized scenario. The approach uses bootstrapping aggregation to create an ensemble of predictive models, leveraging fully connected feed-forward neural networks. These networks are designed with a flexible number of trainable parameters and hidden layers, requiring minimal computational resources. The model is trained on a small set of examples obtained through direct simulations that randomly sample the ODD, bypassing the traditional test matrix definition. Once trained, the bootstrapped model serves as an identity card for the system under test, allowing for continuous performance evaluation across the ODD. The paper demonstrates applications, including safety assessment, failure mode identification, and developing a safe speed recommendation function. The model's compact size ensures rapid execution, facilitating extensive post-analysis for safety argumentation and diagnosis and real-time online use to extend the system's abilities.**

Index Terms—**Highly automated vehicles, predictive models, bootstrapping aggregation, safety assurance, safety assessment, failure analysis, safe speed recommendation, cut-in, logical scenario, ODD.**

I. Introduction

Highly automated vehicles are complex systems. Proving their safe operation within their designated Operational Design Domain (ODD) can be a significant challenge [1]–[3]. Even more challenging is the task of identifying the causes of safety failures and implementing necessary system updates.

This paper presents a method that aids in all three stages: assessment, diagnosis, and update.

A. Cherubini, G. P. Rosati Papini, M. Piazza, and M. Da Lio are with the Department of Industrial Engineering, University of Trento, Trento, Italy (email: antonello.cherubini@unitn.it; gastone.rosatipapini@unitn.it; mauro.dalio@unitn.it; mattia.piazza@unitn.it). A. Plebe is with the Department of Computer Science, University College London, London, United Kingdom. (email: a.plebe@ucl.ac.uk)

Manuscript received on June 13, 2024. This work was supported by the European Commission under Grant Horizon Europe 101069573 (SUNRISE), by the EU-funded program FSE REACT EU, D.M.1062 by the Italian Ministry of Education and by the European Commission under Grant H2020 731593 (Dreams4Cars). The Associate Editor for this paper was .... (*Corresponding Author: Antonello Cherubini*)

A. Contribution

The method's core involves developing a stochastic model capable of predicting safety outcomes (collision, near-miss, or safe state) within a given parametrized scenario (logical scenario), including bounds for uncertainty. This approach broadly falls under the category of bootstrapping aggregation [4], [5]: we construct an ensemble of predictive models (learners) through bootstrapping [6], [7] and combine their outputs to estimate outcomes with bounded uncertainties.

The individual learners are constructed using fully connected feed-forward neural networks due to their flexibility in adjusting the number of trainable parameters and hidden layers. A few neurons and parameters—typically in the tens and hundreds, respectively—are sufficient.

The model is trained using a reasonably small number of examples obtained through direct simulations that randomly sample the logical scenario. This replaces the traditional approach of defining a test matrix, thereby decoupling the model from subsequent uses.

Once trained, the bootstrapped model is a comprehensive assessment tool for the system under test, allowing continuous evaluation of system performance across the logical scenario. We demonstrate its effectiveness through three applications:

- Safety assessment against various hypothetical risks for three self-driving systems.
- Support in identifying different failure modes.
- Development of a safe speed recommendation function.

During inference, the small size of the model components enables rapid execution, facilitating numerous post-analysis tasks such as safety argumentation and diagnosis. Moreover, the model is fast enough for online usage during system operation to address diagnosed limitations, such as providing safe speed recommendations.

B. Structure of the paper

This paper is structured as follows: Section II provides a literature overview about safety assessments for automated vehicles. Section III mentions related international initiatives. Section IV describes the proposed methodology and Section V applies the methodology to a case study. Section VI analyzes the trained models. Section VII discusses three applications. Finally, Section VIII gives some important remarks. Additional details about network optimization are in Appendix A.

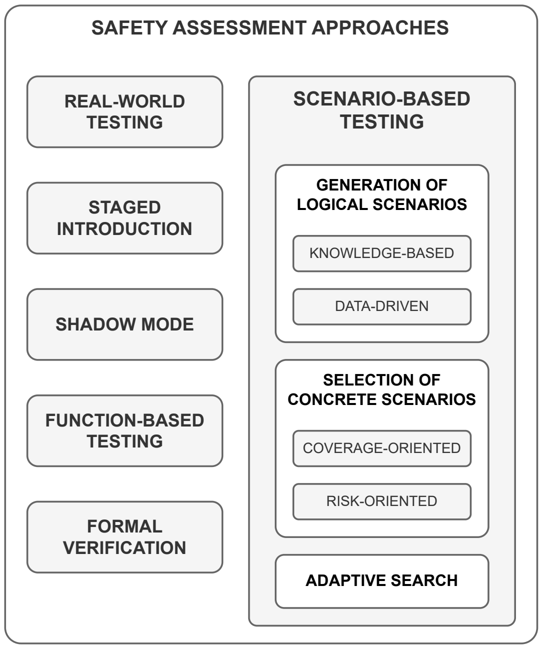


Fig. 1. A possible taxonomy of the most adopted strategies for safety assessment of AVs.

II. Safety Assessment in the Literature

Due to the strong interest in rapidly introducing automated vehicles (AVs) to the market, a significant number of publications on their safety validation have emerged. However, there is still no consensus on a unified approach to assessing AV safety.

Drawing from the work of Riedmaier et al. [8], we identify six primary categories of safety assessment, as shown in Fig. 1."

1) *Real-world testing.* Also called Naturalistic-Field Operational Testing (N-FOT) would ideally prove the safety of the system by test-driving the AV in real traffic, observing its performance, and making statistical comparisons to human drivers. However, Kalra and Paddock [1] demonstrated that an AV must drive 11 billion failure-free miles to determine with significant confidence that its failure rate is lower than the human driver fatality rate in the US. Even with a fleet of 100 AVs driving 24 hours a day, 365 days a year, at an average speed of 25 mph, this would take 518 years. Aside from the logistical and economic impracticalities, a solely distance-based safety evaluation has another significant downside. Under naturalistic traffic conditions, there is an extremely low level of exposure to high-risk events, which are crucial for demonstrating the system's safety under challenging conditions.
2) *Staged introduction.* This approach works by artificially limiting the ODD of the system to make real-world testing economically feasible and not overly time-consuming. For example, the ODD can be restricted to driving on a specific section of a road under good visibility. If the vehicle is assessed as safe in this limited ODD, the scope can be gradually expanded. System manufacturers such as Daimler and Bosch are applying this procedure by testing their Level 4 systems in a delimited road network in San Jose, California[1]. Waymo at Phoenix, San Francisco and Los Angeles[2] In practice, however, this approach only defers testing to the later extensions of the Operational Design Domain (ODD).
3) *Shadow mode.* Employed by manufacturers such as Tesla[3], this approach involves operating the autonomous driving agent passively in a vehicle under human control. While the autonomous agent receives real sensor inputs, it does not control the vehicle's actuators. Simulations are then conducted to assess the agent's decisions by comparing them to the human's behavior. However, the validity of these simulations is limited. The behavior of other road users is influenced by the decisions of the actual driver. If the autonomous agent makes different decisions, the subsequent actions of other users become inconsistent.
4) *Function-based testing.* This approach focuses on testing a specific functionality within a system, as defined by a set of requirements. A small set of fixed tests is designed based on these requirements. While this method is commonly used to assess the safety of Advanced Driver Assistance Systems (ADAS), such as Adaptive Cruise Control[4] and Advanced Emergency Braking Systems[5], it encounters challenges when applied to Autonomous Vehicles (AVs). Defining the required functionality of AVs in every conceivable situation proves to be very difficult.
5) *Formal verification.* This approach employs mathematical methods to formally prove the safety of systems across the entire Operational Design Domain (ODD). A critical step in these methods is formalizing traffic rules into a machine-readable format. Some techniques focus on proving that the AV cannot cause any accidents for which it is at fault [9]. Others aim to determine the states a system can reach from given initial states, inputs, and parameters [10]. If the reachable set of the AV does not intersect with the predicted occupancy sets of other traffic participants, the AV is deemed safe. Another formal verification approach involves synthesizing correct-by-construction controllers [11], which are automatically generated from formal specifications. However, the main drawback of all these methods is their current lack of scalability to complex systems and their computational expense (and, often, the assumption of correct behavior

[1]https://www.bosch-presse.de/pressportal/de/en/bosch-and-mercedes-benz-start-san-jose-pilot-project-for-automated-ridesharing-service-204032.html accessed on June 10th 2024

[2]https://support.google.com/waymo/answer/9059119?hl=en accessed on June 10th 2024

[3]https://www.youtube.com/live/Ucp0TTmvqOE at 2:56:00, accessed on June 10th 2024

[4]ISO 15622:2018 Performance requirements and test procedures https://www.iso.org/standard/71515.html accessed on June 10th 2024

[5]Regulation No 131 of the Economic Commission for Europe of the United Nations (UN/ECE) http://data.europa.eu/eli/reg/2014/131/oj accessed on June 10th 2024

for the other road participants).

6) *Scenario-based testing:* This widely adopted approach assesses the safety of AVs through simulations of various scenarios, each defined by a set of parameters. However, as scenarios become more complex, the need for additional parameters arises, resulting in a computational burden in enumerating and simulating these scenarios. Therefore, a critical aspect of scenario-based testing lies in the design and selection of scenarios that align with test requirements and expose the most relevant high-risk events.

The work presented in this paper falls within the category of scenario-based testing. The rest of this section analyzes diverse characteristics of scenario-based approaches.

A. Scenario-based testing

An essential step in scenario-based testing is how to define the scenarios. Menzel et al. [12] distinguish three types of scenarios according to their level of abstraction:

(i) *functional scenarios* are defined by natural language descriptions, e.g., a three-lane motorway in a curve;
(ii) *logical scenarios* are the parameterization of functional scenarios and define the ranges and distributions of the parameter spaces, e.g., lane width ($w \in [2.3,\ 3.5]\ m$), curve radius ($r \in [0.6,\ 0.9]\ km$);
(iii) *concrete scenarios* represents an instance of a logical scenario and is defined by precise parameter values, e.g., $w = 3.2\ m$, $r = 0.7\ km$.

The ODD of an AV is typically defined in terms of functional scenarios.

When assessing the AV's safety within the ODD, it is necessary to convert these functional scenarios into logical scenarios. A primary challenge involves determining the most appropriate parameters, along with their respective ranges and distributions. Furthermore, to conduct the tests, specific concrete scenarios must be selected from the logical scenarios.

1) Generation of logical scenarios: The challenge of generating a comprehensive set of logical scenarios has led to diverse strategies. Consider a common case study in ODD analysis, such as a cut-in scenario, where an external vehicle enters the lane of the ego vehicle, forcing the latter to react and adjust its trajectory. To parameterize cut-in scenarios, a broad range of parameter choices can be found in the literature. For example, [13] and [14] employ two different sets of five parameters to describe the scenario, where the former includes the road width, while the latter considers the relative velocity. In contrast, [2], [3], [15] examine a six-parameter scenario with three vehicles. Moreover, [16] and [17] include polynomial parameters to model the longitudinal and lateral motion, resulting in a total of eight and fifteen parameters, respectively. In other words, modeling a functional scenario with logical scenarios involves projecting from an (almost) infinite-dimensional space into a relatively low-dimensional parametrized space that will never fully represent the nuances of the real world.

According to Riedmaier et al. [8], strategies for generating logical scenarios can be classified into two groups:

(i) *knowledge-based generation*, which creates logical scenarios by leveraging information from experts, standards, and guidelines, which are in some cases structured as ontologies [18];
(ii) *data-driven generation*, which extracts information directly from real-world driving data, typically using machine learning [19], Monte Carlo [20], or Importance Sampling [21].

A fundamental prerequisite for data-driven scenario generation is the comprehensive nature of the naturalistic dataset. While many automotive companies possess proprietary datasets, some large-scale datasets have been made publicly accessible in recent years, such as [22], [23]. However, in naturalistic datasets, the occurrence of highly relevant situations (accidents, near misses) is naturally low. Hence, data-driven approaches must address this issue by applying sophisticated strategies to augment the data or accelerate the discovery of rare events, ensuring the creation of a sufficient set of relevant, high-risk scenarios [24]. On the other hand, knowledge-based approaches can address this challenge by deliberately designing logical scenarios with parameter ranges and distributions that lead to high-risk situations.

2) Selection of concrete scenarios: The second challenge is determining how to select concrete scenarios for the safety assessment of the AV system. The goal is to identify a limited, representative set of concrete scenarios to draw reliable conclusions about the AV's performance [3]. Selection approaches are divided into two main categories:

(i) *coverage-oriented selection*, which aims to maximize testing coverage under a certain coverage metric, thereby testing as thoroughly as possible[6];
(ii) *risk-oriented selection*, which generates targeted scenarios to facilitate fault detection, focusing on high-risk events, collisions, and worst-case scenarios.

A straightforward method for coverage-oriented selection is to create a "test matrix" by sampling from the parameter ranges, producing a table where each row represents a concrete scenario. The test matrix is considered a credible and repeatable method [25]. For example, the Euro NCAP's test protocol for autonomous emergency braking (AEB) relies on a test matrix[7]. However, enumerating all possible concrete scenarios through a test matrix can be intractable, and additional strategies are needed to focus on high-relevance scenarios within the matrix. Another method for coverage-oriented selection is the T*-wise* coverage [26], widely adopted in software and communication systems verification. This method assumes that T-parameter combinations can trigger most system circumstances, and that the values of other parameters are weakly related to the results. T-wise coverage represents the ratio of distinct T-parameter combinations covered during testing to the total number of possible T-parameter combinations.

[6]M. Tatar (2018), *Chasing critical situations in large parameter spaces.* https://www.pegasusprojekt.de/en/lectures-publications

[7]https://cdn.euroncap.com/media/26996/euro-ncap-aeb-c2c-test-protocol-v20.pdf accessed June 10th 2024

However, a key challenge with this method is determining the empirical evidence for the values of T, which is currently not feasible for AV testing.

A valid alternative to coverage-oriented selection is to explore unusual, dangerous, and extremely critical scenarios. Risk-oriented selection is also called *falsification* as it seeks to find counterexamples that violate the safety requirements. Sun et al. [3] classify these methods into four categories depending on the focus of the selection.

(i) *High-risk scenarios* are situations with the potential to endanger vehicles and road users, possibly resulting in severe accidents if they were to happen in reality. Identifying high-risk scenarios can be achieved through various methods, such as binary search techniques [27], evolutionary algorithms [28], or by amplifying the criticality of typical traffic scenarios using nonlinear optimization methods [29]. However, there is still no uniform standard for defining high-risk scenarios.

(ii) *Boundary scenarios* are situations located in regions of the parameter space where small changes lead to transitions between safety and danger. For example, Tuncali et al. [30] design a robustness evaluation function to identify boundaries between safe and unsafe behavior. In another study [31], they introduce a framework for the adversarial generation of test scenarios by detecting collisions in boundary regions of the parameter space.

(iii) *Worst-case scenarios* represent the most critical situations an AV may encounter within its Operational Design Domain (ODD). Various methods are used to identify worst-case scenarios in the literature, including genetic algorithms [32] and neural networks [33], [34]. A common critique of analyzing worst-case scenarios is their perceived improbability, which may make accounting for them too costly under typical operating conditions. However, despite their rarity, these scenarios remain physically possible, and evaluating an AV in such conditions is not futile. For example, Liu et al. [35] propose a dual formulation that considers both the most likely worst-case scenario and a less likely one. Furthermore, assessing worst-case performance can help reinforce public trust in AVs.

(iv) *Collision scenarios* are the focus of a substantial portion of research, including the work presented in this paper. Methods for searching for collisions include rapidly-exploring random trees [36], rule-based approaches [37], and generative neural networks [38]. Additionally, Calò et al. [39] employ genetic algorithms to explicitly search for collisions that can be avoided by changing the parameters of the path planner.

B. *Adaptive search with Surrogate models*

Scenario-based testing typically involves simulating all scenarios upfront. However, an increasingly popular alternative is the adaptive search, also called the Adaptive Design of Experiments (ADOE) method. Instead of examining all concrete scenarios simultaneously, the adaptive search begins with a limited set and gradually identifies new ones for the next simulations. A surrogate model (SM) is fitted at every step to approximate the AV's behavior in the simulated scenarios. The SM is then used at each iteration to select the next concrete scenario to test. This selection process is guided by an acquisition function, which balances the selection of potentially dangerous scenarios with exploring rare ones. These values are computed by the SM for each candidate scenario.

Adaptive search is particularly effective in accurately determining the boundary between safe and dangerous scenarios. Bhosekar and Ierapetritou [40] emphasize the importance of selecting the most appropriate surrogate model for the performance of adaptive search, which can vary depending on the application. Sun et al. [2] compare six popular SMs and find that extreme gradient boosting performs best across different Operational Design Domains (ODDs).

C. *Result evaluation*

The taxonomy presented thus far encompasses diverse methodologies for testing AV systems. The test results obtained need to be evaluated to draw meaningful conclusions about the safety of the AV. Junietz [41] identifies two types of evaluations: *microscopic evaluation* focuses on individual scenarios, while *macroscopic evaluation* refers to a statistical statement on the risk of the system, e.g. the occurrence rate of fatal accidents. Scenario-based testing provides evaluations at the microscopic level. The problem of how to exploit microscopic evaluations to obtain a macroscopic evaluation of the AV is still an open question.

Regarding microscopic evaluation, Mahmud et al. [42] provide an overview of the most adopted criticality metrics. They distinguish among indicators based on temporal proximity, spatial proximity, and deceleration rate.

III. Safety assessment in international initiatives

Several industrial research programs have recently received funding, reflecting the critical importance of safety assessment for the market introduction of automated vehicles. The currently ongoing projects include the European SUNRISE (Safety Assurance Framework for Connected and Automated Mobility Systems), the German VVM (V&V Methods), the Japanese SAKURA, the Korean KATRI, and the American AVSC (Automated Vehicle Safety Consortium).

These initiatives employ a combination of methodologies outlined in the previous section to establish standardized safety assessment procedures.

For instance, SUNRISE[8] is organized into three macroblocks. The first focuses on creating scenarios based on the Operational Design Domain (ODD) input, storing them in machine-readable formats. It utilizes both knowledge-based and data-driven generation methods. The framework aims to integrate diverse and heterogeneous sources of information. In the second macroblock, concrete scenarios are created and allocated to various test instances, including simulations, X-in-the-loop (XIL), and proving grounds. The final macroblock involves the evaluation of test results at the micro-scale and

[8]https://ccam-sunrise-project.eu/about/ accessed June 10th, 2024

their aggregation into an analysis to draw macroscopic conclusions. If deemed insufficient, strategies such as generating new concrete scenarios, reallocating to different test instances (e.g., from simulation to proving ground), or requesting completely new logical scenarios can be implemented. Additionally, an in-service monitoring and reporting system is envisioned to provide lifelong support.

IV. Methodology

In alignment with Section I-A, our objective is to develop a fast model capable of continuously predicting an intelligent vehicle's performance across a defined logical scenario while also providing quantified uncertainty.

Let $\boldsymbol{x}$ be a vector collecting a concrete scenario parameters p_i —i.e., $\boldsymbol{x} = \{p_1, ..., p_n\}$. We can think of $\boldsymbol{x}$ as a point in the logical scenario parameter space. Let:

$$\boldsymbol{e} = \boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

indicate the mapping between $\boldsymbol{x}$ and the outcome $\boldsymbol{e}$, which is a vector of the performance indicators obtained by testing the concrete scenario $\boldsymbol{x}$.

The function $\boldsymbol{f}(.)$ can be evaluated with experiments on a few points $\boldsymbol{x}_i$, or with a validated simulation tool on a much larger number of points. In the latter case, $\boldsymbol{f}(.)$ is typically deterministic: repeating a simulation yields the same results unless noise and/or hidden parameters are explicitly modeled in the simulation environment. We consider here the simulation deterministic case.

Even though simulations are much faster than real-world tests, the number of simulations $\boldsymbol{f}(\boldsymbol{x}_i)$ necessary to assess a system's performance across a given logical scenario may still be prohibitive [2], [3]. Furthermore, evaluating a single $\boldsymbol{f}(\boldsymbol{x}_i)$ via simulation is hardly compatible with computations carried out *during system operation*. While simulations can be performed offline to evaluate a system's performance, they cannot be executed online (during operation) to assess whether the current condition is safe and determine the necessary actions to achieve a safer condition. This latter capability is crucial because a self-driving system with this ability could be aware of its own limits and environmental risks, enabling it to engage in cautious preventive behaviors.

We aim to develop a predictive model that is significantly faster than simulations (as demonstrated later, achieving near-million-fold acceleration in computations is feasible):

$$\{\boldsymbol{e}\} = \tilde{\boldsymbol{f}}(\boldsymbol{x}) \tag{2}$$

The tilde sign in $\tilde{\boldsymbol{f}}(.)$ indicates that (2) approximates test outcomes (1). The vector notation $\{.\}$ denotes that $\tilde{\boldsymbol{f}}(.)$ is an ensemble model that estimates the distribution of $\boldsymbol{e}$, taking into account uncertainties arising from finite training sets. Therefore, a crucial aspect of the method is quantifying the approximation errors to determine the confidence with which conclusions derived using (2) are valid for real test or simulation results (1).

A. Implementing the approximant function $\tilde{\boldsymbol{f}}(.)$

There are several machine learning methods potentially suited for creating approximant functions $\tilde{\boldsymbol{f}}(.)$. In this paper, we opt for neural networks due to their flexibility. By adjusting the number of neurons, we can easily manage the number of trainable parameters to align with the available training examples. Varying the depth of the networks (i.e., distributing the neurons across multiple hidden layers) and selecting different activation functions allow us to control the complexity of the approximant function.

Appendix A provides an analysis of various network architectures. For the case study presented in this paper, which comprises 5000 input-output ($\boldsymbol{x} \rightarrow \boldsymbol{e}$) examples, with 3000 used for training and the remaining for validation and testing, we identified an optimal configuration. This configuration involves two hidden layers with Tanh activation, totaling 30 neurons and 343 trainable parameters. Notably, this parameter count is approximately 1/10 of the number of examples in the training set. Specifically, the network is labeled "10 Tanh 20 Tanh", indicating two hidden layers with 10 and 20 neurons, respectively, both utilizing Tanh activation, and a final softmax layer for classification.

B. Creating an ensemble model via bootstrapping

The bootstrapping technique is a method used to estimate the variance of statistical estimators in regression or classifier models [6]. This method involves creating numerous alternative training datasets by resampling from an existing dataset. These new datasets are generated in such a way that they can be considered as being drawn from the same population as the original dataset. By training a model on the resampled datasets, an empirical distribution of estimators is obtained. A tutorial on the application of bootstrapping to neural models, which also discusses the relevant theory, can be found in [7].

There are several variants of the bootstrapping method. In this paper, we use the Split/Train variant [7, Section 6.1]. Given N examples, we randomly divide them into three subsets of N', N'', and N''' elements, ensuring no overlap between them. The N' subset is used as the training set for updating the weights in the gradient descent algorithm. The N'' subset serves as the validation set for model selection

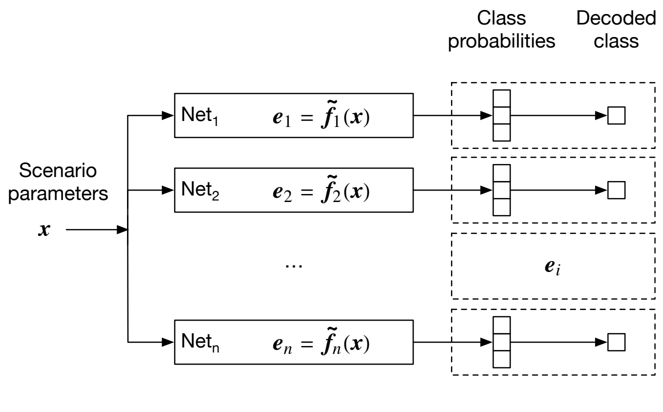

Class probabilities
Decoded class
Scenario parameters
$\boldsymbol{x}$
Net$_1$
$\boldsymbol{e}_1 = \tilde{\boldsymbol{f}}_1(\boldsymbol{x})$
Net$_2$
$\boldsymbol{e}_2 = \tilde{\boldsymbol{f}}_2(\boldsymbol{x})$
...
$\boldsymbol{e}_i$
Net$_n$
$\boldsymbol{e}_n = \tilde{\boldsymbol{f}}_n(\boldsymbol{x})$

Fig. 2. Boostrapped Neural Model.

and overfitting monitoring during training. The N''' subset is reserved as the test set to evaluate the model's performance on independent examples not used during training or model selection. Further details are given in Appendix A.

By splitting the N examples into different training, validation, and test subsets K times, we train K equally plausible neural network models, which can be arranged in parallel as shown in Fig. 2. The outputs of the $\tilde{\boldsymbol{f}}_{\boldsymbol{k}}(.)$ models form an empirical distribution of predictions.

Since the number of network parameters is on the order of hundreds (Section IV-A), repeating the training of one network K times in the bootstrapping process is feasible. On average, training one network takes less than 40 seconds on an Apple Silicon M1 using the ARM-optimized version 12.3.1 of Wolfram Mathematica. Therefore, training, say, $K = 100$ models, takes just over an hour.

C. Decision (aggregation) algorithms

As the bootstrapping method produces an ensemble of predictive models (Fig. 2), the question arises of how to combine the empirical distributions of predictions $\boldsymbol{e}_i$. This falls within the realm of ensemble learning [4], [5], and specifically for this paper, bagging (Bootstrap Aggregation) techniques.

Referring to Fig. 2, we consider three different methods: one using class probabilities and two using decoded classes.

1) Mean probabilities: This method computes the average of all learners' probabilities for each class. For example, the collision probability of the ensemble model $\bar{p}_c(\boldsymbol{x})$ is:

$$\bar{p}_c(\boldsymbol{x}) = \frac{1}{K}\sum_{i=1}^{K} p_{c,i}(\boldsymbol{x}) \tag{3}$$

where $p_{c,i}(\boldsymbol{x})$ is the collision probability of model i. The mean probabilities $\bar{p}_c(\boldsymbol{x})$, $\bar{p}_n(\boldsymbol{x})$ and $\bar{p}_s(\boldsymbol{x})$ are then used to decode the ensemble output class (where s, n and s stand for collision, near-miss and safe outcome).

2) Majority voting: The class with the majority of votes is the ensemble output class. In the case of *ex aequo*, preference goes to collision and near-miss in the order.

3) Consensus threshold (n_0): This method is introduced to bias the ensemble classifier towards minimizing misclassifications of the collision class. The algorithm counts the output classes, and if the collision class reaches a threshold of votes (n_0), the output is set to collision. Otherwise, the output follows the majority voting scheme.

The rationale behind this algorithm is that if only a few classifiers return collision, then the risk of collision should not be dismissed. Thus, we set a threshold n_0/K to indicate the requested confidence for not misclassifying collisions. By doing this, we improve the Recall of the collision class at the expense of its Precision.

The output class can be formalized as follows:

$$e = \begin{cases} \text{collision } (c) & n_c \geq n_0 \\ \text{near-miss } (n) & n_c < n_0 \wedge n_n \geq n_s \\ \text{safe } (s) & n_c > n_0 \wedge n_n > n_s \end{cases} \tag{4}$$

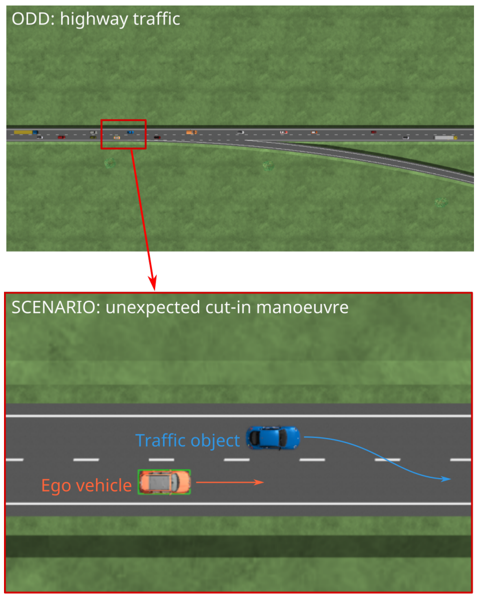


Fig. 3. An example scenario: In an attempt to avoid missing an exit, a distracted driver (in the blue car) suddenly changes lanes. The ego vehicle (orange car) must react quickly to avoid a collision.

where n_c, n_n, and n_s are the number of votes for each class.

In the following example, we will use this algorithm with two thresholds: $n_0/K = 1/100$ (confidence 99%) and $n_0/K = 10/100$ (confidence 90%).

D. Creating the original training examples

Given a logical scenario with parameter ranges, we sample the scenario to create input-output examples $x_i \rightarrow e_i$, $i \in [1, N]$. The sampling points x_i are initially spread uniformly and randomly in the domain of the logical scenario to avoid biases and create a training set balanced across the logical scenario. A simulation is conducted for each x_i to calculate the corresponding output e_i. In other words, the pairs $x_i \rightarrow e_i$ are formed with (1).

The number N of samples is initially determined based on a predefined computing time budget. However, since the model quantifies uncertainties, we can assess whether additional samples might be necessary.

V. Case study

In Fig. 3, an example is presented to illustrate the method. The Operational Design Domain in this case is a two-lane motorway. The scenario involves a distracted driver in the second lane attempting a careless cut-in maneuver to avoid missing an exit. These types of situations, which can lead to accidents, are not uncommon and can be found documented online, for example in [43].

To analyze this case, we will follow the steps outlined in Fig. 4. The first three steps correspond to the scheme described in Section II-A.

Specifically, the logical scenario is shown in Fig. 5. It is characterized by 5 parameters: the velocities of the ego and obstacle (v_0 and v_1), the leading distance (d), the duration of

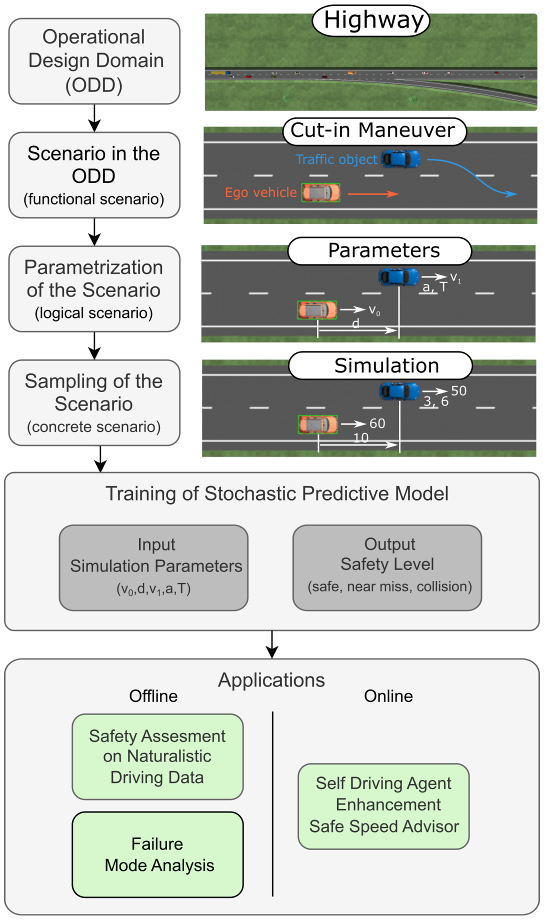

Operational Design Domain (ODD)
Highway
Scenario in the ODD (functional scenario)
Cut-in Maneuver
Traffic object
Ego vehicle
Parametrization of the Scenario (logical scenario)
Parameters
a, T
v1
v0
d
Sampling of the Scenario (concrete scenario)
Simulation
50
3, 6
60
10
Training of Stochastic Predictive Model
Input Simulation Parameters (v0,d,v1,a,T)
Output Safety Level (safe, near miss, collision)
Applications
Offline
Online
Safety Assesment on Naturalistic Driving Data
Failure Mode Analysis
Self Driving Agent Enhancement Safe Speed Advisor

Fig. 4. Workflow followed for safety and failure mode analysis and for the safe speed function of the case study.

the lane change (T), and the obstacle's (average) acceleration during the lane change (a), which determines the final velocity $v_{1,f} = v_1 + aT$.

This parametrization is used to explain the methodology. We do not claim that is it optimal. In the literature cut-in scenarios have been described with different sets of parameters (ranging from 5 to 15) as discussed in Section II-A1.

We should keep in mind that during the parametrization process, some aspects of the functional scenario are inevitably lost in the parametric representation. For instance, while T represents the time taken to complete the lane change, it does not fully describe the exact trajectory shape. Additionally, the assumption of average longitudinal acceleration for the entire duration T, used to represent decelerations such as those seen in the referenced video [43], is a simplified model. Furthermore, factors such as lane widths, road curvature, vehicle dimensions, vehicle type, surrounding traffic, and other real-world conditions are not accounted for in the model.

A. Training the stochastic predictive model

After clarifying the above limitations (the format of a logical scenario is not the topic of this paper), our contribution begins at the step corresponding to the concrete scenario (Fig. 4). At this point, we deviate from the workflow described in

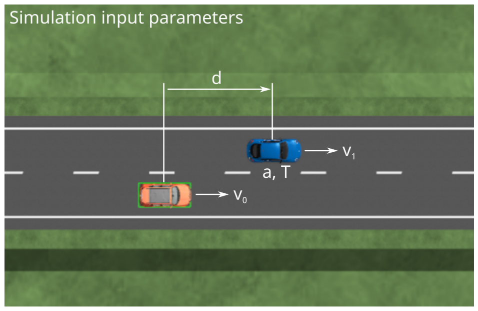


Fig. 5. The logical scenario is defined with 5 parameters: ego and obstacle velocities (v_0 and v_1), leading distance (d), lane change duration (T), and obstacle acceleration during the lane change (a).

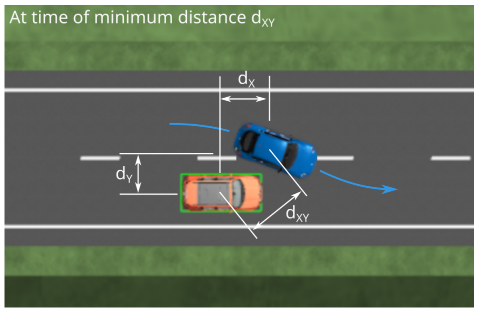


Fig. 6. Simulation output: For each simulation, we record the position of the obstacle (blue car) in the ego reference frame when the Euclidean distance between the cars, d_{XY}, is at a minimum, or at the moment of collision.

Section II-A2 as follows.

1) Sampling the logical scenario: we sample the logical scenario according to Section IV-D with $N = 5000$ samples, which proves to be sufficient for the following applications.

We can define an indicator of sampling density as follows:

$$N_1 = N^{1/D} \tag{5}$$

where D is the number of dimensions of the logical scenario. In our case, $D = 5$, $N = 5000$, and hence $N_1 = 5.5$. This means that the density of the 5000 samples is, on average, the same as a regular grid with 5.5 hypothetical divisions per dimension (which defines the resolution of the model and is commented on later in Section VI-E).

The simulations were performed using IPG CarMaker [44]. Each simulation represents a hypothetical event lasting 60 seconds. Fortunately, the simulations run at a faster pace than real-time. With the Intel i7 12700H CPU, which we utilized, they typically run 13 times faster. Therefore, each event took about 4.5 seconds to be simulated, in addition to another 2.5 seconds of overhead. Consequently, the total time required to produce the 5000 training examples was approximately 10 hours.

The simulations start with the ego vehicle (the orange car in Fig. 5) beginning in the right lane at an initial velocity v_0. The obstacle (blue car) starts with a velocity of v_1 in the left lane at a distance d (Fig. 5). At the start of the simulation, the obstacle

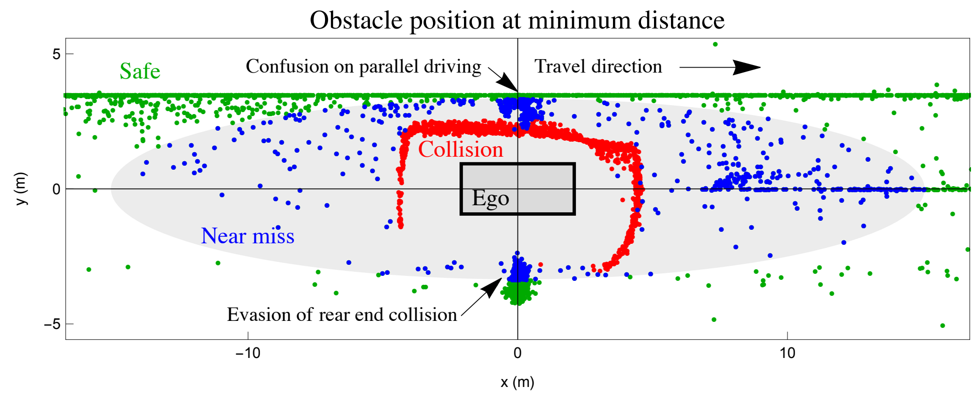


Fig. 7. Codriver training set. Collision points in red and the nearest points in blue or green. A near-miss is registered if the nearest point falls within the elliptic grey zone.

initiates a lane change maneuver lasting for a time T (the exact trajectory of the obstacle is determined by its controller, which is the IPG Driver). During the lane-changing maneuver, the blue car may also experience a constant longitudinal acceleration or deceleration of a.

The ranges for the logical scenario parameters are outlined in Table I. It's important to note that the values for T and a go beyond the typical naturalistic driving ranges. Additionally, we are assuming that the cut-in maneuver might initiate even if the obstacle is behind the ego vehicle ($d < 0$), although this is highly unlikely. The aim at this stage is to build a model that remains valid outside the scope of naturalistic driving scenarios, encompassing potential but improbable situations. This will allow us to use the trained model later on to assess risks that fall within the training domain, which includes natural driving data. For certain combinations of T, a, and v_1, the final velocity of the obstacle ($v_{1,f} = v_1 + aT$) might be negative. If this happens, T is adjusted to make $v_{1,f} = 0$, i.e., the obstacle stops when it reaches the right lane.

TABLE I
RANGES OF VARIABLES IN THE LOGICAL SCENARIO (Ω)

Variable		**Range**	
		min	**max**
v_0	(m/s)	10.0	25.0
$v_1 - v_0$	(m/s)	-5.0	7.0
T	(s)	1.5	7.5
a	(m/s^2)	-6.0	3.0
d	(m)	-20.0	40.0

2) Systems under test (SUTs): We will illustrate the methodology described in this paper using three self-driving agents. Two of these agents, *Codriver* [45] and *Motion Planner* [46], are systems that control both longitudinal and lateral dynamics. They were developed in previous projects. The third agent is a simple longitudinal controller based on the Intelligent Driver Model (IDM). It has been designed specifically for this study, following a logic similar to [47]. More specifically:

- The *Codriver* is an autonomous driving agent inspired by biological concepts such as the Affordance Competition hypothesis [48]. It was initially developed as part of the EU H2020 Dreams4Cars project[9] and has since undergone further improvements. This system is capable of producing emergent behaviors and adaptive behaviors to handle unexpected situations without being explicitly programmed to do so [49].
- The *Motion Planner* is an advanced robotic trajectory planner developed as part of a recent project. The system combines sampled-based tree exploration (semi-structured RRT*) and analytic optimal tree connections (clothoids and minimum-time-minimum-jerk motion primitives) to generate safe and efficient trajectories in complex environments. It can also be adapted to produce different driving styles [46].
- The *Intelligent Driver Model* (IDM) is the well-known, simple follow-the-leader model for longitudinal control [50]. The algorithm controls the longitudinal dynamics in relation to leading vehicles and can be adapted to merging vehicles as shown in [47].

3) Performance indicators: In general, the model output $\boldsymbol{e}$ in (1, 2) can be any vector of performance indicators relevant to the functional scenario's goal. As mentioned earlier, this study aims to assess whether and to what extent the systems under test can avoid a collision when the obstacle maneuvers incorrectly. That is why $\boldsymbol{f}(.)$ in Section IV-C is implemented as a classifier with three possible output classes: 1) collision (c), 2) near-miss (n), and 3) safe clear path (s). We introduced the "near miss" class to gather more informative data than simply collision/non-collision.

For each simulation, we defined the output classes as follows: In the event of a collision, we recorded the position of the obstacle in the ego reference frame at the time of the collision. If there is no collision, we recorded the minimum Euclidean distance (d_{XY}) between the ego vehicle and the obstacle, as well as the position of the obstacle in the ego reference frame at the minimum distance. The origin of the ego reference frame is located in the center of the car, with the longitudinal axis (x) pointing forward and the lateral axis (y) pointing leftward. For example, in Fig. 6, the minimum

[9]http://www.dreams4cars.eu/en

distance d_{XY} is 3.3 m, and the relative coordinates of the obstacle are $(d_X, d_Y) = (2.53, 2.13)$ m.

Figure 7 displays the collision points in red and the nearest points in blue or green for the codriver dataset. The grey ellipse represents a shape that just about clears the centerline of the left lane and has a half-length equal to twice the total length of the two vehicles. If the nearest point falls within the grey zone, it is considered a near-miss.

B. Training data and neural model training

The training data consists of points x_i, as determined in Section IV-D, and the corresponding class labels determined in Section V-A3. In total, there are $N = 5000$ input-output examples.

$$\boldsymbol{x_i} \rightarrow \boldsymbol{e_i}, \quad i \in \{1, \ldots, N\} \tag{6}$$

Using the bootstrapping method outlined in Section IV-B, we then train $K = 100$ neural networks to create the model shown in Fig. 2. The choice of $K = 100$ is justified in Appendix A (see also [7] for general guidelines concerning the numerosity of the models). Training all 100 networks takes around 3700 seconds using an Apple M1 8-core CPU.

To evaluate a single network branch (Fig. 2), the number of operations needed is calculated as follows:

1) The first layer consists of 10 neurons with an input dimension of 5. This requires 10x5 multiplications, 50 additions (including the biases), and 10 Tanh operators.
2) The second layer has 20 neurons, requiring 10x20 multiplications, 200 additions, and 20 Tanh operators.
3) The output layer is composed of 3 neurons, which needs 20x3 multiplications, 60 additions, and 3 softmax operators.

In total, one branch in Fig.2 requires 620 multiplications/additions and 33 operators, i.e., 62000 simple operations plus 3300 functions for the entire $K = 100$ network model.

When evaluations are conducted in batches offline, the inference time, including the overhead of using the MXNet neural network framework, is approximately 32 microseconds on the mentioned M1 processor. This is roughly 200,000 times faster than the 7 seconds required to run a simulation in the IPG environment, which itself is several times faster than real-time.

For online use, the weights of the neural layers can be extracted and included in a custom function that performs the same operations as the network. In this case, the computations will have much less overhead and will be even faster.

VI. Analysis of the trained models

Before exploring different types of applications (Section VII), we will first analyze the learned function in this Section.

In Fig. 8, two bi-dimensional cross-sections of the parameter space are shown at $v_0 = v_1 = 25m/s$. The chart on the top represents an obstacle deceleration of $a = -3m/s^2$; at the bottom, an obstacle proceeding at a constant speed.

The figures display the superimposed output class of the 100 bootstrapped neural models for the three agents. Pure colors indicate that all models predict the same class, while blended colors indicate disagreement.

It's immediately clear that the three systems behave quite differently, and the difference is unambiguously captured.

A. Confidence levels

For a given lane change time, the interval of leading distances that causes collisions is well defined, and the intervals lacking consensus are also clear. Figure 9 is a different representation of the bottom left chart of Fig. 8. It shows the contours of the collision class of the individual models of Fig.2. Dashed contours outline the areas where all the models agree. Additionally, contours representing 50%, 90%, and 99% quantiles (indicating the percentage of models that agree) are also displayed. Having chosen $K = 100$, a granularity of 0.01 is obtained for the quantile contours.

Notably, regions lacking consensus tend to be relatively short in the longitudinal direction d. A couple of meters separates the confidence levels corresponding to 50, 90, and 99 percentiles. This means that safe-distance gaps might be a surrogate means to guarantee collision-free performance.

B. Generalization capacity of the ensemble model

To evaluate the generalization capacity of the ensemble classifiers described in Section IV-C, we prepared a new independent test set consisting of 1613 examples generated from new simulations.

Fig. 10 shows the confusion matrices for the majority voting aggregation algorithm (Section IV-C2) and for the consensus threshold algorithm for respectively $n_0/K = 10/100$ and $n_0/K = 1/100$, i.e., 90 and 99 percentiles (Section IV-C3)[10]

As illustrated in the figure from left to right, the number of false negatives for the collision category decreases as the threshold n_0/K decreases. When $n_0/K = 1/100$, no collision examples are mistakenly classified as near misses or safe events. Hence, the false negative events occur between the 0.5 and 0.99 quantiles contours in Fig. 9.

For safety argumentation, one might want to minimize the amount of collisions that are mistakenly classified as safe. Therefore, the algorithm on the far right might be the most suitable option. However, in this case, several near-miss events and a few safe events are incorrectly classified as collisions, as the confusion matrices show.

The other aggregation criteria in Section IV-C may fulfill different roles that are explained below.

C. Accuracy, Precision and Recall

Table II provides a comparative analysis of the accuracy of the aggregation algorithms mentioned in Section IV-C. Additionally, the table presents the precision and recall for the collision class and the intervals of the same indicators for the individual networks illustrated in Fig.2.

[10]The mean probabilities algorithm (Section IV-C1) is very similar to the majority voting one and is not shown.

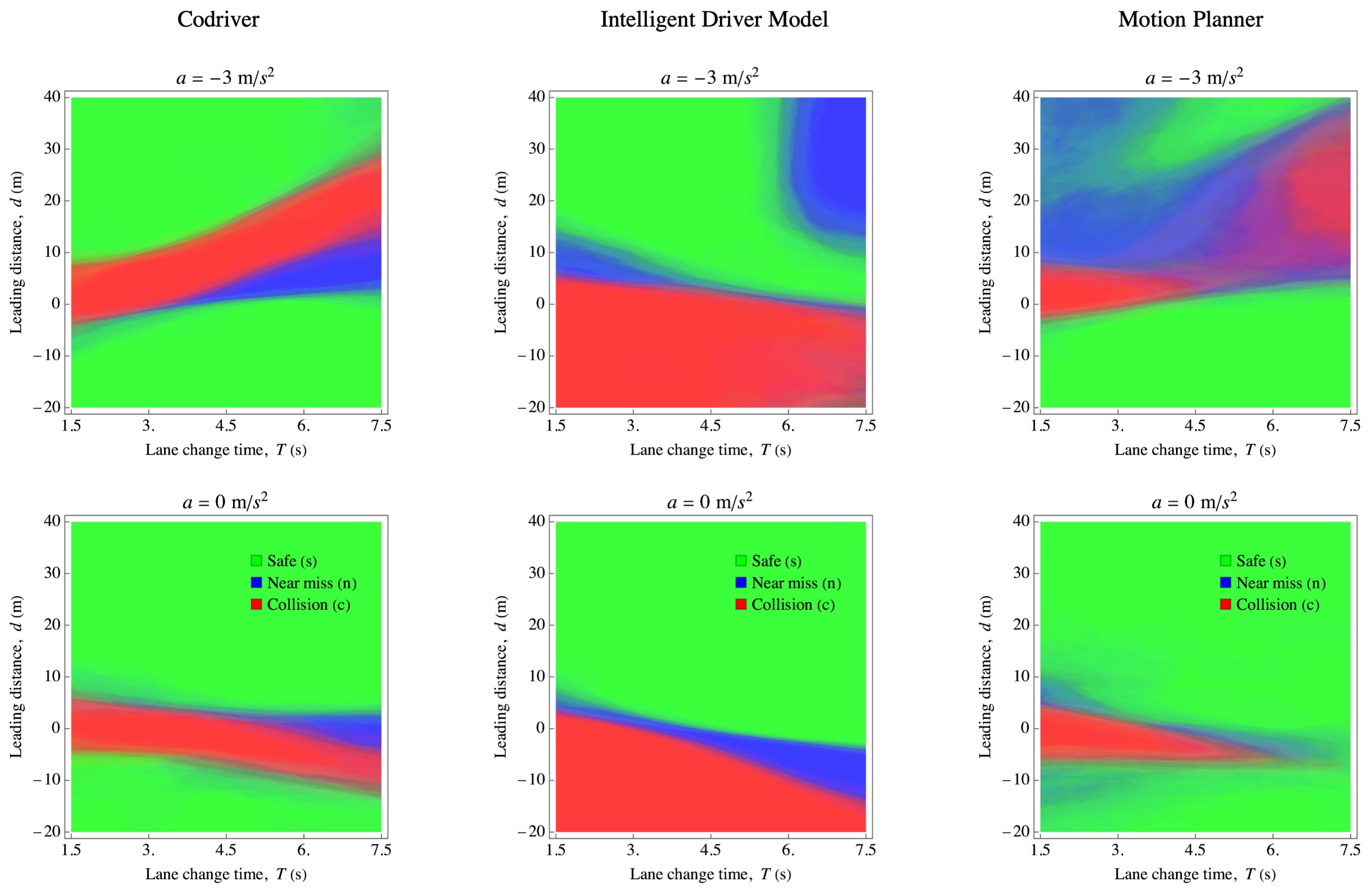


Fig. 8. Cross-sections of the parameter space at $v_0 = v_1 = 25m/s$. The figures display the superimposed output class of the 100 bootstrapped neural models for the co-driver agent. Pure colors indicate agreement, while blended colors indicate disagreement.

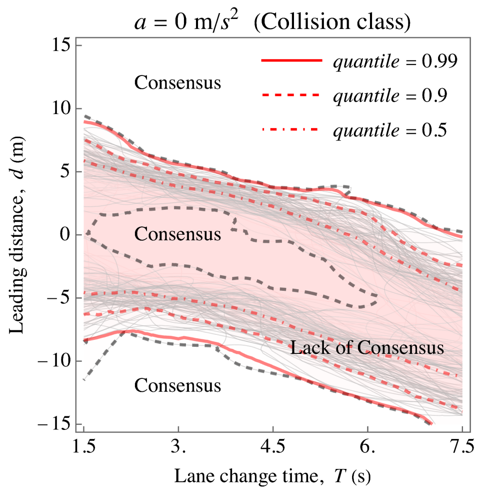


Fig. 9. Contours of the collision class of the individual models. Dashed contours outline the areas where all the models agree. Contours representing 50%, 90%, and 99% quantiles are also displayed.

The mean probabilities criterion demonstrates superior performance over the individual component networks across all the indicators. The majority voting criterion yields equally good results. Moreover, the consensus threshold criteria enhance the recall on the collision class, albeit with a trade-off of reduced precision.

D. *Characterizing and documenting different systems*

Fig. 8 demonstrates the discrepancies between the systems being tested and illustrates how the model (2) can differentiate between various systems in the logical scenario. We can consider (2) as a snapshot of the system, which can be stored for immediate use and future documentation (specifically, to track the system's updates over time).

Some major differences can be explained as follows.

Compared to the other systems, the IDM, which only controls the vehicle's speed and looks at the front of the vehicle, cannot avoid rear-end collisions ($d < 0$). On the other hand, the other systems can avoid such collisions by changing lanes.

Upon closer examination, the IDM seems to be more effective at avoiding front-end collisions, but this is because it does not have a realistic model of the longitudinal controller that slows the vehicle's response.

For the Codriver and the Motion Planner, when the obstacle's deceleration is $a = -3m/s^2$, the collision interval shifts forward as the time T taken for a lane change increases. This is because the deceleration affects the entire lane change

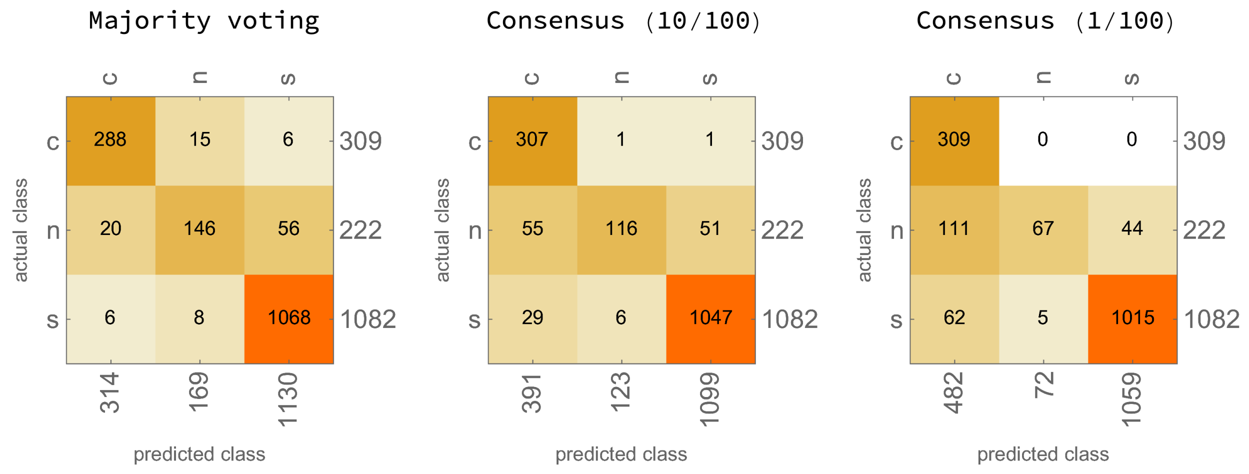


Fig. 10. Codriver new test set. Confusion matrices for the majority voting aggregation algorithm (Section IV-C2) and for the consensus threshold algorithm for respectively $n_0/K = 10/100$ and $n_0/K = 1/100$

	Individual networks Mean (SD)	Mean probabilities	Majority voting	Consensus ($n_0/K = 10/100$)	Consensus ($n_0/K = 1/100$)
Accuracy	0.912 (0.008)	**0.931**	**0.931**	0.911	0.862
Recall (collision)	0.901 (0.019)	0.929	0.932	0.994	**1.**
Precision (collision)	0.895 (0.018)	**0.917**	**0.917**	0.785	0.641

TABLE II
PERFORMANCE INDICATORS OF INDIVIDUAL NETWORKS AND ENSEMBLE MODELS

time, resulting in a greater overall reduction of the obstacle's speed[11].

Also, in the top row, the Motion Planner has more near-misses for forward collisions due to less efficient prediction of obstacle paths compared to the Codriver.

When the obstacle acceleration is zero (bottom row), the interval in which collisions can occur shifts backward and becomes smaller as the time taken for a lane change increases. In this scenario, a larger value of T allows more time for the system to slow down and prevent hitting the obstacle from behind.

The best-performing agent may vary across different parts of the scenario. For instance, when the obstacle moves at a constant speed (bottom row), the Motion Planner is better at avoiding close rear-end collisions. This can be explained by the fact that the Codriver is strictly prohibited from exceeding the speed limit and brakes when a close obstacle completely inhibits its "motor space" [45, Section II-C-5].

E. Caveat

The modeling technique described so far relies on the ability of neural networks to generalize to new data. However, it's important to use a sufficient number of training examples to avoid losing model resolution. Equation (5) defines the average density of the sampling points used to build the model, indicating the model's ability to resolve details. For example, if there's a small region that isn't sampled, the model will overlook it. This could also occur if the critical region is just beyond the edge of the logical scenario.

[11]One potential objection is that this result occurs because we use the parameter T to represent the duration of both the deceleration and the lane change. If we were to use a separate parameter, such as T_a, to model the deceleration time, we would be separating the speed variation from the lane change time. However, this approach would require the use of a 6-dimensional logical scenario.

More realistically, if an interesting region receives very few samples, the trained networks will tend to smooth and overlook a few samples, depending on the regularization technique used in training. To check if this is happening, we can monitor where the misclassified points occur. If they are in areas of full agreement, it suggests insufficient resolution.

If we don't know where the critical regions are, uniform sampling is the best choice (as we did). If we have expert or data-driven knowledge, then a non-uniform sampling of the logical scenario can be used. We can also refine the sampling where insufficient resolution is found.

VII. Applications

The stochastic model developed in this paper has various applications (latest block of Fig. 4), including offline and online uses. Offline applications are carried out *while the system is not in operation* and involve system evaluation and malfunction diagnosis. Online applications are carried out *in parallel with system's operation* and focus on supervising system operation, monitoring risks, and providing preventive safety recommendations. We describe here the three possible applications introduced in Fig. 4: Safety assessment (VII-A), Failure Analysis (VII-B) and Safe Speed Advisor as an example of Self Driving Agent Enhancements (VII-C).

A. Safety assessment for hypothetical risks

The ensemble model (2) describes the performance of a system across a given logical scenario Ω (Table I), regardless

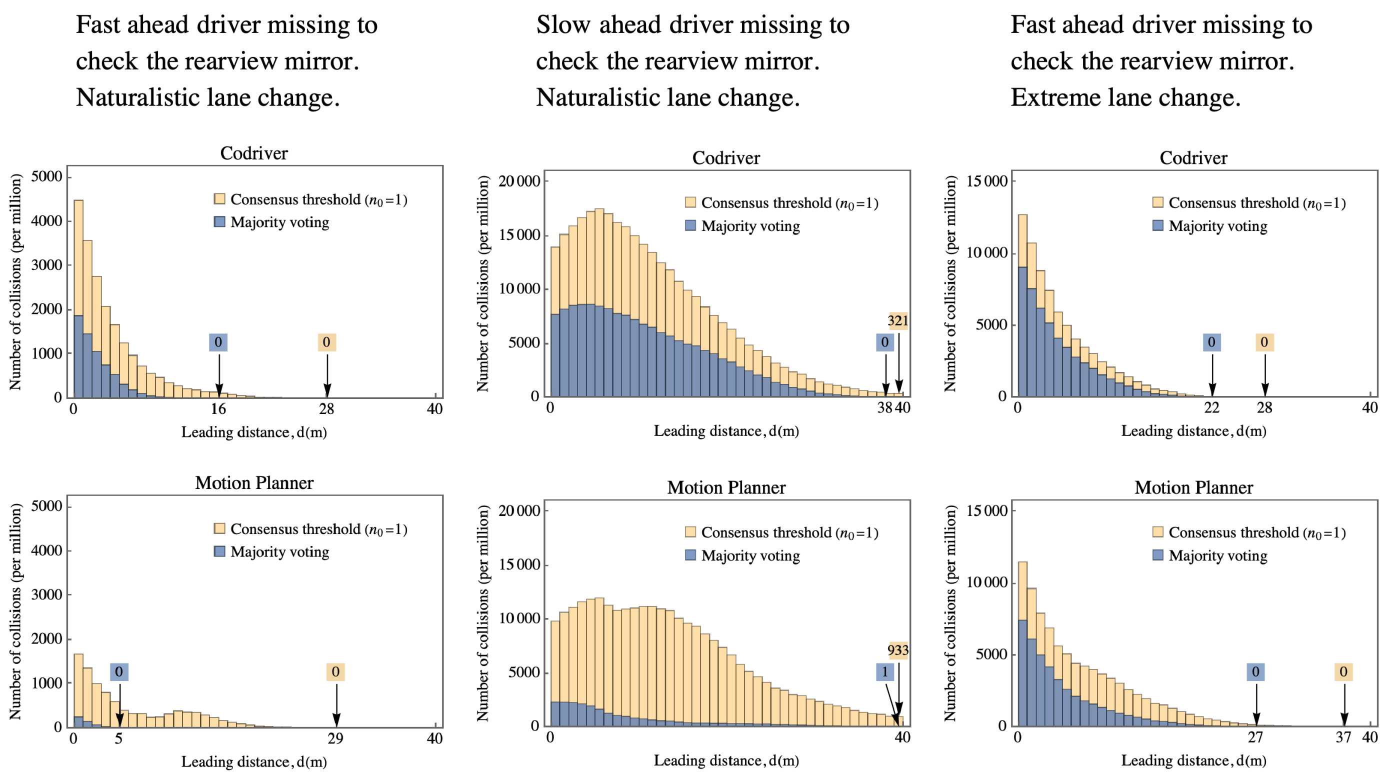


Fig. 11. Risk assessment under three hypotheses: 1) The driver fails to check the rearview mirror, travels ahead and faster than the ego, and changes lanes with naturalistic parameters. 2) Similar to hypothesis 1, except the driver travels slower than the ego. 3) Similar to hypothesis 1, but with extreme lane change parameters. The most alarming condition arises when the merging vehicle travels slower than the ego.

of the probability of points $\boldsymbol{x} \in \Omega$ to occur in driving situations.

The model (2) can be subsequently used to assess a system's performance in relation to various hypothetical risks or events.

A hypothetical risk i can be defined as a subset of the logical scenario $\Omega_i \subset \Omega$, which represents conditions that could occur under certain hypotheses, such as, e.g., misbehavior by another road user. To fully describe risk i, we need to know the probability of a specific condition $\boldsymbol{x} \in \Omega_i$ to occur ($p_i(\boldsymbol{x}), \boldsymbol{x} \in \Omega_i$); for instance, derived from naturalistic driving data or observed situations.

If $p_i(\boldsymbol{x})$ is available, the system's performance, for example the probability of collision $p_{c,i}$, can be computed as follows:

$$p_{c,i} = \int_{\Omega_i} p_i(\boldsymbol{x}) p_c(\boldsymbol{x})\, d\boldsymbol{x} \tag{7}$$

where $p_c(\boldsymbol{x})$ indicates the probability that the state $\boldsymbol{x}$ causes a collision: $p_c(\boldsymbol{x})$ is evaluated with one of the aggregation criteria given in Section IV-C. For example, using consensus threshold (4):

$$p_c(\boldsymbol{x}) = \begin{cases} 1 & n_c(\boldsymbol{x}) \geq n_0 \\ 0 & n_c(\boldsymbol{x}) < n_0 \end{cases} \tag{8}$$

where $n_c(\boldsymbol{x})$ is the number of bootstrapped models predicting collision (Fig. 2).

Other definitions are possible. For example, $p_c(\boldsymbol{x}) = n_c(\boldsymbol{x})/K$ would define the probability of collision to be proportional to the number of votes. However, (8) incorporates error margins defined by threshold n_0/K and may thus be preferred.

Integration via Monte Carlo sampling: We may encounter inefficiency when using a numerical integration algorithm for the majority voting (IV-C2) and the consensus threshold (IV-C3) aggregation criteria due to the discontinuous nature of the function $p_c(\boldsymbol{x})$. To address this, we can estimate the integral in equation (7) using the Monte Carlo method. In this approach, we generate a large number N of sample points $\boldsymbol{x}_i$ that follow the distribution $p_i(\boldsymbol{x})$, denoted as $\boldsymbol{x}_i \sim p_i(\boldsymbol{x})$, and then proceed to count the number of collisions evaluating $p_c(\boldsymbol{x_i})$ with the model, e.g., with (8):

$$p_{c,i} \approx \frac{1}{N} \sum_{i=1}^{N} p_c(\boldsymbol{x_i}) \tag{9}$$

This approach is feasible because the calculation of batches of $p_c(\boldsymbol{x_i})$ with the model (2) is very rapid, as previously mentioned (Section IV-B).

The convergence of (9) can be easily tracked as N increases. In practical terms, the computation of (9) with $N = 1,000,000$ takes approximately 32 seconds on an Apple Silicon M1 processor, and the estimated summation stabilizes after the initial 10.000 points.

Dealing with unknown distributions: Frequently, we do not have the probability distribution $p_i(\boldsymbol{x})$ available. However, we can still use the information about Ω_i to determine if Ω_i is free from collisions with a certain level of confidence. One way to do this is by generating a uniform distribution of sample points $\boldsymbol{x}_i$. The outcomes from equation (9) will then indicate the portion of Ω_i that is not collision-free.

We present three examples showcasing the flexibility of model (2) in assessing various hypothetical risks.

Example 1. Driver missing to check the rearview mirror, ahead and faster than ego, with naturalistic lane change parameters: In this scenario, the hypothetical risk involves a driver making a sudden lane change without being aware of his/her surroundings, for instance, if he/she fails to check the rearview mirror. However, aside from this mistake, we assume that the driver changes lanes with naturalistic driving parameters. Hence, the subdomain Ω_1 may be delimited as follows:

- v_0 spans the whole range given in Table I;
- $v_1 \geq v_0$ (a slower obstacle will be another example), i.e., $v_1 - v_0 \in [0, 7]m/s$;
- $d > 0$, i.e., the obstacle is ahead (we ditch the cases where the driver initiates the lane change behind ego because he/she would be aware of ego);
- the lane change time T follows naturalistic driving distributions. More precisely, [51, Table 4] suggests that the median lane change time is 5.1 s and that 85% of all lane changes last more than 4 s. Hence, we set $T > 4s$.
- the deceleration a follows naturalistic driving distributions. More precisely, [52, Table 4] suggests that 0.9 quantiles of longitudinal acceleration (for speed range $40 - 70km/h$) happen in the interval $-1.72, 1.13m/s^2$. Hence, we set $a \in [-1.72, 1.13]m/s^2$

After sampling Ω_1 with 1 million uniformly distributed points, we obtained the histogram on the left column of Fig. 11, which shows the collision count for distance bins of 1 meter. Collisions are counted according to the majority vote and the consensus threshold ($n_0 = 1$) criteria. Collisions may or may not happen within the same distance bin depending on the values of the other four logical scenario parameters (the number of cases per bin is 40.00, but there are much fewer collisions per bin). The Codriver agent has higher collision rates because the longitudinal control is limited in jerk. Conversely, the motion planner is quicker at changing the longitudinal acceleration. Despite different evasive capacities, they reach zero collision rate at the same distance.

Example 2. Same as example 1, but with a slower obstacle: In this example, the hypothetical risk is the same as in Example 1, but the obstacle's velocity is slower than the ego vehicle. Ω_2 is the same as Ω_1, except that $v_1 - v_0 \in [-5, 0]m/s$. The corresponding collision counts are shown in the second column of Fig.11. Compared to the first column, it is clear that a slower obstacle poses a higher risk. Furthermore, the maximum danger occurs some meters ahead (closer obstacles, being slower, may pass behind the ego vehicle).

It is important to note that the Motion Planner exhibits a bimodal behavior and does not achieve zero collision rate up to 40 m leading distance. This is consistent with the reduced safe regions observed in Fig. 8 (top, right). This occurs because the Codriver has a better prediction of obstacle trajectories when they are not moving at a constant speed.

Example 3. As in example 1, but with the obstacle performing extreme lane change maneuvers: In this scenario, the hypothetical risk is similar to that in Example 1, except that the lane change duration is in the range of $T \in [2, 4]s$ (the 15 percentiles not included in Example 1, [51, Table 4]), and the longitudinal acceleration spans a broader range of $a \in [-4.77, 3.25]m/s^2$ [52, Table 4]. Compared to the first column of Fig. 11, using higher accelerations or faster lane changes is less dangerous than when the obstacle is already slow. The safe distance is not significantly affected by the Codriver and is slightly affected by the Motion planner.

The three examples above demonstrate how the modeling technique developed in this paper can be utilized to formulate various hypotheses about potential risks, understand the factors that have the greatest impact on safety, and highlight the different performances of various systems (which could be the same system at different stages of development). Moreover, breaking the risks down into individual components reveals that a system's performance is more complex than can be expressed with a single number and varies depending on the perspective from which the system is being considered.

B. Failure analysis via exploration of the collision boundary

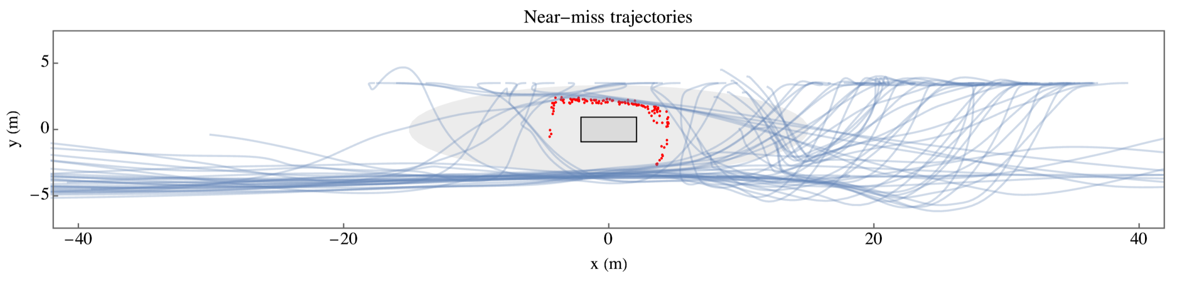
In previous examples, we have seen how to evaluate a system's performance in various risk scenarios. However, there may be situations where designers need to provide explanations for system failures or determine if certain changes can enhance performance. This type of analysis may involve a detailed, step-by-step offline review of the sequence of events and decisions within the systems and is typically conducted in a limited number of cases. The model (2) can assist in identifying and prioritizing the most significant cases and monitoring the system's progress following updates.

The workflow is as follows:

- Identify a reduced number of points regularly spaced on the boundary of the collision regions.
- Simulate these points and cluster the resulting trajectories.
- Analyze one example per cluster (the same failure mode likely causes similar trajectories).
- After the system's update, train a new model for documentation and check for regions where the system improved and areas where the system worsened.

In the following example, we present the first two steps for the Codriver agent.

Sampling the boundary of collision regions: The boundary of collision regions is a 4-dimensional manifold. We can easily find points close to the boundary by randomly sampling the logical scenario. Since the computation of model (2) is extremely fast, we can evaluate millions of points without the need for more sophisticated algorithms.



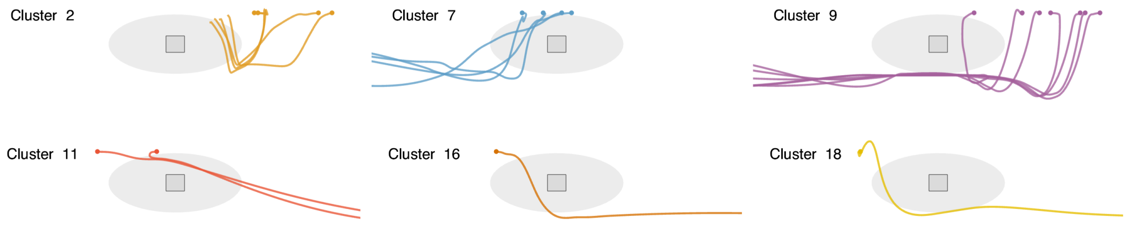


Fig. 12. Failure mode analysis: The logical scenario is sampled using model 2 to identify near-miss events close to the collision boundary. These events are then clustered to reveal a small number of potential failure modes that can be replayed offline by designers.

To find candidates lying close to the boundary, we use the decision criterion in IV-C1. The collision class is characterized by $\bar{p}_c > \max(\bar{p}_n, \bar{p}_s)$, meaning that the aggregated probability of collision $\bar{p}_c$ is the largest one.

The collision boundary is defined by $\bar{p}_c - \max(\bar{p}_n, \bar{p}_s) = 0$. Thus, the samples $\boldsymbol{x}_i$ are sorted according to the function $f(\boldsymbol{x}) = |\bar{p}_c(\boldsymbol{x}) - \max(\bar{p}_n(\boldsymbol{x}), \bar{p}_s(\boldsymbol{x}))|$. In the example given here, we sampled the logical scenario with $N_1 = 1,000,000$ points and retained $N_2 = 2000$ points, where $-0.011 < f(\boldsymbol{x}_i) < 0.011$.

We still have too many candidates, and they are not evenly distributed on the boundary. There are more points where the gradient of $f(\boldsymbol{x})$ is smaller and fewer where the gradient is higher. So, we use the k-means algorithm to cluster the N_2 points around common centroids. We specify the number of clusters as $N_3 = 200$, and then take the corresponding centroid, effectively reducing the number of candidates by tenfold. The sample size is hence reduced to $N_3 = 200$ points, which are now more evenly distributed near the boundary. After simulations, among these initial points, 57 produce a nearly-miss event, and we focus on them as they are more informative than the collisions. The trajectories in an ego reference frame are shown in the top image of Fig. 12.

Clustering the near-miss trajectories: To categorize different types of failures, near-miss trajectories can be grouped using the mean-shift clustering algorithm, resulting in 19 different clusters, of which the most noteworthy are depicted in Figure 12 at the bottom.

A replay of one example per cluster gives the following insights. Cluster 2 involves a scenario where the merging vehicle comes to a complete stop, blocking the Ego vehicle between the obstacle and the right lane edge. Clusters 7 and 9 showcase the ego vehicle changing lanes either in front of or behind the obstacle, whose final position is on the right. In cluster 11, a quick-moving obstacle narrowly passes the Ego vehicle on the left. Clusters 16 and 18 illustrate two potential rear-end collisions, which are avoided by the Codriver's lane change maneuver.

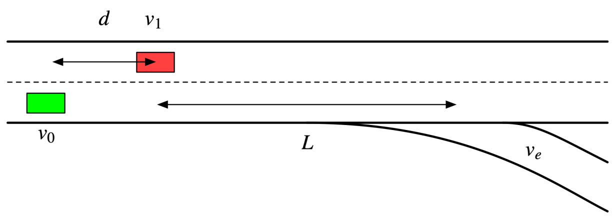


Fig. 13. Exit scenario

C. Safe Speed Advisor

The model (2) is fast enough to be used in online applications. For instance, it can assess the safety of the current state and, if the current state is unsafe, suggest a nearby safe state. In online applications, the logical scenario parameters are generally well-known.

Let's take the scenario shown in Figure 13 as an example. The velocities of the obstacle (v_1), the ego vehicle (v_0), and the distance to the obstacle (d) are known. By considering the risks outlined in examples VII-A and VII-A, for instance, the possibility of the obstacle driver to take an exit without being aware of the ego vehicle, it is feasible to estimate plausible values for the lane change duration (T) and deceleration (a) based on the obstacle velocity (v_1), the exit ramp speed limit (v_e), and the distance to the exit ramp (L). This enables the assessment of whether the current state is safe in relation to the hypothetical risk.

If the current state is not safe, we can recalculate by adjusting the ego vehicle's velocity to find a safe speed. Figure

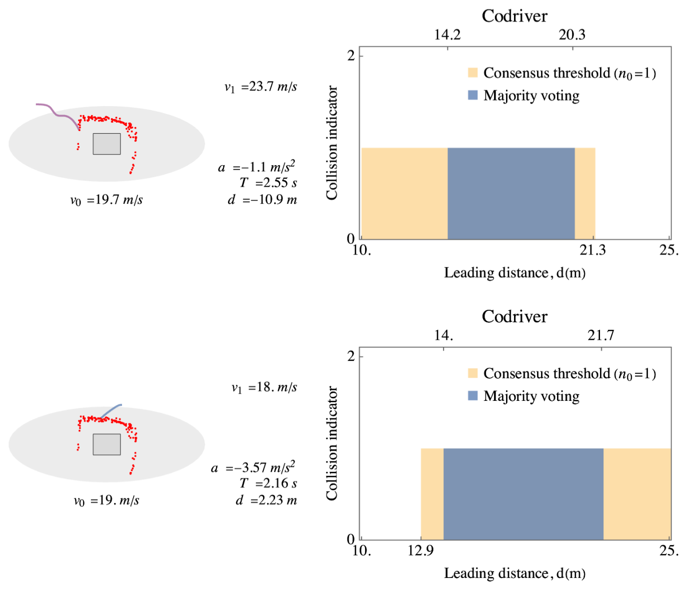


Fig. 14. Online speed recommendation: Safety is evaluated with (2) to discover safe speed intervals considering the current situation.

14 provides two examples (extracted from the collision events of Section VII-C).

In the top example, the obstacle is faster, resulting in a rear-end collision. The chart on the right illustrates the collision prediction using two aggregation criteria. Importantly, to be safe for rear-end risks, the ego vehicle should be traveling at a speed *exceeding* $21.3m/s$.

In the bottom example, the obstacle is slower. In this case, safe speeds are those below a certain threshold ($12.9m/s$).

VIII. Conclusions

The approach described in this paper involves capturing a snapshot of a system within a specific logical scenario. Using bootstrapping aggregation, we develop a model that tracks its accuracy. This model can be applied to online and offline applications, as demonstrated by three examples. It can also be used to monitor the progress of system design, ensuring that updates lead to improvements without introducing new failure modes.

Regarding safety assessment on a broader scale, we would like to comment on some key points. Firstly, it needs to be clarified how many functional and logical scenarios are necessary to cover a complex operational design domain (ODD) comprehensively. According to Kalra, billions of miles must be driven to evaluate a vehicle's safety statistically. Thus, the question arises - how many functional and logical scenarios are required for safety assessment? Secondly, translating a functional scenario into various logical scenarios implies that a finite-parameter and finite-symbol scheme is used to represent an infinite number of possibilities. This approach implicitly subscribes to the Cartesian model (representations with amodal symbols), effectively testing safety within a subset of reality. The limitations of the Cartesian model in relation to situations that cannot be described within the given parameters and symbols are discussed in [49]. Thirdly, an essential aspect of safety argumentation is coverage analysis. The model 2 operates as a continuous function across the logical scenario and can be leveraged, with the caveat in Section VI-E, to assess system performance throughout the entire logical scenario. However, the issue of whether a functional scenario is adequately covered by a collection of logical scenarios remains. To address this, we could apply the modeling methodology from this paper to alternative logical scenarios to determine if alternative parameters are better or necessary. Nevertheless, this increases the number of scenarios to be studied and conflicts with Kalra's argumentation. Additionally, we cannot confirm whether any logical scenarios have been omitted at this stage, as this can only be addressed at the level of (data-driven) scenario generation.

IX. Contribution of the Authors

Antonello Cherubini developed the Case Study (Section V) and the initial version of the analysis of the trained model (Section VI), which included a simplified three-dimensional logical scenario. Gastone Pietro Rosati Papini studied the optimal neural architecture (Appendix A) and contributed to the analysis of its performance (Section VI). Alice Plebe studied the State of the Art (Section II), which included evaluating the performance of literature surrogate models on the logical scenario presented in this paper. Mattia Piazza developed the IDM and the Motion Planner agents (the Codriver was pre-existing). Mauro Da Lio is the group leader. He contributed to the methodology (Section IV), the final version of the analysis of the trained model (Section VI), and the applications (Section VII).

The authors collectively discussed the study, and all shared a common opinion on the positions expressed.

Mauro Da Lio carried out the final text editing. Grammarly was used for proofreading and for improving clarity.

Appendix A
Neural Network Architecture and Hyperparameter Optimization

This appendix explains our decisions regarding the implementation of neural networks and the specific training methods. We begin by discussing the rationale behind selecting the dataset size and then proceed to determine: i) The structure of the neural network and the most appropriate learning rate (Appendix A-A), ii) How to properly divide the dataset into training, validation, and test sets (Appendix A-B), iii) The selection of the number of models for the bootstrapping strategy (Appendix A-C).

The dataset size derives from allocating one full day for simulations that resulted in 5000 simulations. Subsequently, following the considerations explained in Section VI-E, we found that this number adequately covers all conditions of the logical scenario.

A. Structure of the network and learning rate

In order to optimize the predictive abilities of the networks in the specific operational setting, we studied several fully connected neural architectures. We varied the number of layers, the number of neurons in each layer, and the activation function to find the best combination.

Additionally, we studied the learning rate, a parameter that significantly affects the network's performance.

The optimization process involves two rounds: in the first round, we determine the optimal learning rate and the most promising shallow structure. In the second round, we studied architectures with varying numbers of hidden layers.

The selection of the network architecture was based on the Akaike Information Criterion (AIC) [53]. The Akaike Information Criterion penalizes models based on their degrees of freedom, indicating the model with the best trade-off between predictive capacity and a minimum number of parameters.

We decided to use ADAM as the optimizer for training because of its adaptive learning rate. We used a batch size of 64 to introduce a good level of randomness while still allowing for quick training. The number of epochs was set to 5000 to make sure that the model converges, but we also set a stopping criterion of 200 epochs if no improvement is observed in the validation set. The loss function used was the cross-entropy. Each network was trained three times with random initialization to ensure the reliability of the AIC evaluation.

Figure 15 shows the (natural logarithm) Log(AIC) values obtained for the two rounds of optimization for all the considered network architectures and learning rates. Lower AIC values indicate better performance.

In the figure, each network is labeled with hidden layers, which are the only ones that vary. All the considered networks have the same last layers, consisting of 3 neurons and a softmax activation function, which allows for generating a probability distribution across the three output classes considered.

In **Round 1**, the shallow structure with the lowest AIC is the "40 Tanh", trained using a learning rate of 0.001. The final network selected in **Round 2** has the structure "20 Tanh 10 Tanh".

B. Train-validation-test split methodology

In this section, we determine the best ratio for dividing the dataset for training and validation while keeping the test set at 20% (maintaining this size for the test set helps us evaluate performance more reliably, as reducing the test set further can make the evaluation unstable). Hence, we considered different partition ratios: 50:30:20 (train:validation:test), 55:25:20, 60:20:20, 65:15:20, 70:10:20, and 75:5:20. For each of these, we trained the network selected in Round 2, called "20 Tanh 10 Tanh", 10 times to calculate the average accuracy and its variance. The same stopping criterion used in the previous section was applied to the validation set.

The results show that as the size of the training set increases, the accuracy remains constant, but the variance of the accuracy in the validation set increases. A high variance in the validation set can negatively affect the overall performance. Therefore, we choose the 60:20:20 proportions.

C. Number of bootstrapped models and model robustness

With the above selected network structure and learning rate ("10 Tanh 20 Tanh", 0.001), we trained 150 networks with 60:20:20 data partition ratio. The training followed the bootstrapping strategy known as "split and train" [7].

A stochastic model, similar to the one shown in Fig. 2, was created by randomly selecting (with repetitions) a number, K, ranging from 5 to 150, from a total of 150 networks. This process was repeated 10 times for each value of K. The mean and standard deviation across the K networks were then calculated for the three classes probabilities. In Fig. 16, the standard deviation (the mean of the standard deviation of the three classes) is displayed as K increases. The BoxWhisker chart illustrates the variation in the ten different extractions. With increasing K, the output variability becomes more stable. A value of $K = 100$ was selected as a good compromise for the number of bootstrapped models.

References

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[2] J. Sun, H. Zhou, H. Xi, H. Zhang, and Y. Tian, "Adaptive design of experiments for safety evaluation of automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 497–14 508, 2021.

[3] J. Sun, H. Zhang, H. Zhou, R. Yu, and Y. Tian, "Scenario-based test automation for highly automated vehicles: A review and paving the way for systematic safety assurance," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 9, pp. 14 088–14 103, 2021.

[4] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, Apr. 2020.

[5] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022.

[6] B. Efron and R. Tibshirani, *An introduction to the bootstrap*. New York: Chapman & Hall, 1994.

[7] U. Michelucci and F. Venturini, "Estimating neural network's performance with bootstrap: A tutorial," *Machine Learning and Knowledge Extraction*, vol. 3, no. 2, pp. 357–373, 2021. [Online]. Available: https://www.mdpi.com/2504-4990/3/2/18

[8] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE access*, vol. 8, pp. 87 456–87 477, 2020.

[9] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[10] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[11] B. Johnson, F. Havlak, H. Kress-Gazit, and M. Campbell, "Experimental evaluation and formal analysis of high-level tasks with dynamic obstacle anticipation on a full-sized autonomous vehicle," *Journal of Field Robotics*, vol. 34, no. 5, pp. 897–911, 2017.

[12] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1821–1827.

[13] Y. Wang, R. Yu, S. Qiu, J. Sun, and H. Farah, "Safety performance boundary identification of highly automated vehicles: A surrogate model-based gradient descent searching approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23 809–23 820, 2022.

[14] S. Thal, H. Znamiec, R. Henze, H. Nakamura, H. Imanaga, J. Antona-Makoshi, N. Uchida, and S. Taniguchi, "Incorporating safety relevance and realistic parameter combinations in test-case generation for automated driving safety assessment," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[15] H. Zhang, J. Sun, and Y. Tian, "Accelerated safety testing for highly automated vehicles: Application and capability comparison of surrogate models," *IEEE Transactions on Intelligent Vehicles*, 2023.

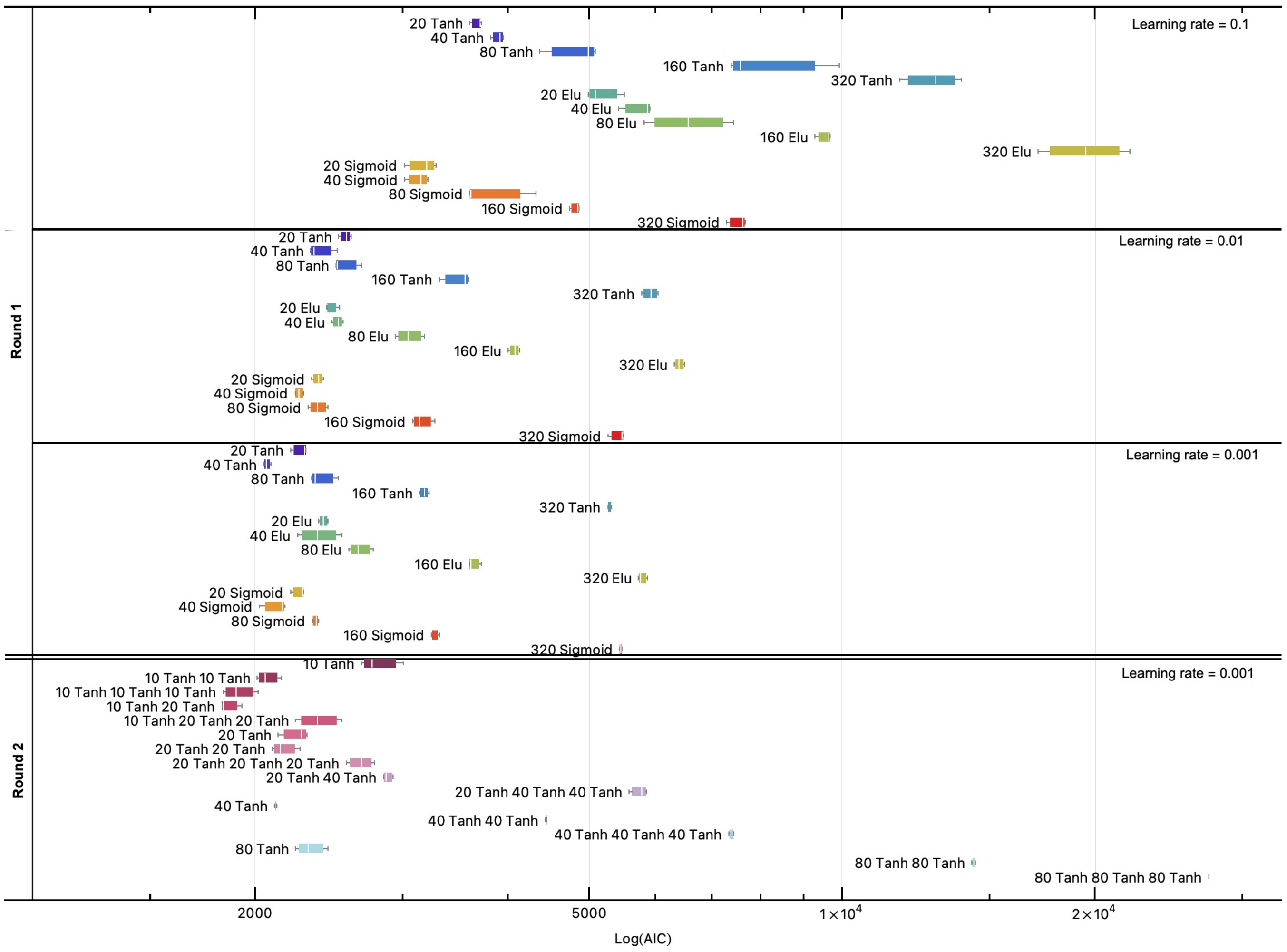


Fig. 15. Value of Log(AIC) for all the examined network architectures and learning rates. Round 1 analyzes the learning rate and the shallow architectures. Round 2 analyzes deep architectures. The chosen network is the 10 Tanh 20 Tanh.

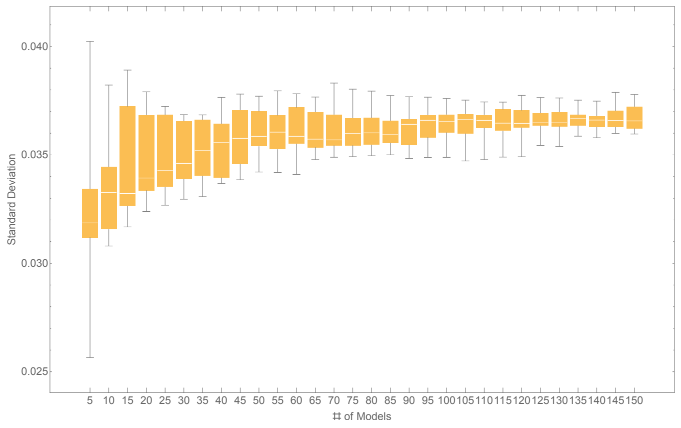


Fig. 16. Box whisker plot of the standard deviation of the class probabilities in a model with K networks. As the number of bootstrapped models K increases, the distribution of probabilities estimated by model (2), Fig. 2, becomes stable.

[16] S. Zhang, H. Peng, D. Zhao, and H. E. Tseng, "Accelerated evaluation of autonomous vehicles in the lane change scenario based on subset simulation technique," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3935–3940.

[17] D. Åsljung, C. Zandén, J. Fredriksson, and M. K. Vakilzadeh, "On automated vehicle collision risk estimation using threat metrics in subset simulation," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 58–63.

[18] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1813–1820.

[19] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein, "Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2383–2390.

[20] G. De Nicolao, A. Ferrara, and L. Giacomini, "Onboard sensor-based collision risk assessment to improve pedestrians' safety," *IEEE transactions on vehicular technology*, vol. 56, no. 5, pp. 2405–2413, 2007.

[21] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 3, pp. 595–607, 2016.

[22] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2118–2125.

[23] A. Blatt, J. Pierowicz, M. Flanigan, P.-S. Lin, A. Kourtellis, C. Lee, P. Jovanis, J. Jenness, M. Wilaby, J. Campbell *et al.*, "Naturalistic driving

study: Field data collection," Transportation Research Board, Tech. Rep., 2015.

[24] I. R. Jenkins, L. O. Gee, A. Knauss, H. Yin, and J. Schroeder, "Accident scenario generation with recurrent neural networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3340–3345.

[25] Y. Xu, Y. Zou, and J. Sun, "Accelerated testing for automated vehicles safety evaluation in cut-in scenarios based on importance sampling, genetic algorithm and simulation applications," *Journal of intelligent and connected vehicles*, vol. 1, no. 1, pp. 28–38, 2018.

[26] C. Amersbach and H. Winner, "Defining required and feasible test coverage for scenario-based validation of highly automated vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 425–430.

[27] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1326–1333.

[28] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2352–2358.

[29] M. Klischat, E. I. Liu, F. Holtke, and M. Althoff, "Scenario factory: Creating safety-critical traffic scenarios for automated vehicles," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[30] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing s-taliro as an automatic test generation framework for autonomous vehicles," in *2016 ieee 19th international conference on intelligent transportation systems (itsc)*. IEEE, 2016, pp. 1470–1475.

[31] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1555–1562.

[32] S. Srikanthakumar and W.-H. Chen, "Worst-case analysis of moving obstacle avoidance systems for unmanned vehicles," *Robotica*, vol. 33, no. 4, pp. 807–827, 2015.

[33] N. E. Chelbi, D. Gingras, and C. Sauvageau, "Worst-case scenarios identification approach for the evaluation of advanced driver assistance systems in intelligent/autonomous vehicles under multiple conditions," *Journal of Intelligent Transportation Systems*, vol. 26, no. 3, pp. 284–310, 2022.

[34] L. Xu, C. Zhang, Y. Liu, L. Wang, and L. Li, "Worst perception scenario search for autonomous driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1702–1707.

[35] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Improving the robustness of an mpc-based obstacle avoidance algorithm to parametric uncertainty using worst-case scenarios," *Vehicle System Dynamics*, vol. 57, no. 6, pp. 874–913, 2019.

[36] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 2879–2886.

[37] S. Masuda, H. Nakamura, and K. Kajitani, "Rule-based searching for collision test cases of autonomous vehicles simulation," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1088–1095, 2018.

[38] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2243–2250.

[39] A. Calò, P. Arcaini, S. Ali, F. Hauer, and F. Ishikawa, "Generating avoidable collision scenarios for testing autonomous driving systems," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE, 2020, pp. 375–386.

[40] A. Bhosekar and M. Ierapetritou, "Advances in surrogate based modeling, feasibility analysis, and optimization: A review," *Computers & Chemical Engineering*, vol. 108, pp. 250–267, 2018.

[41] P. M. Junietz, "Microscopic and macroscopic risk metrics for the safety validation of automated driving," Ph.D. dissertation, Technische Universität Darmstadt, 2019.

[42] S. S. Mahmud, L. Ferreira, M. S. Hoque, and A. Tavassoli, "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs," *IATSS research*, vol. 41, no. 4, pp. 153–163, 2017.

[43] "Cut in crash recorded by dashcam," May 2024, accessed June 10th 2024. [Online]. Available: https://www.instagram.com/reel/C6Cl15uqH4x/?igsh=MThiZnhtdnBtOHkwMA%3D%3D

[44] "IPG CarMaker, professional simulation solution," *https://ipg-automotive.com/en/products-solutions/software/carmaker/*, last accessed Jan 2024.

[45] M. Da Lio, R. Donà, G. P. Rosati Papini, and K. Gurney, "Agent architecture for adaptive behaviors in autonomous driving," *IEEE Access*, vol. 8, pp. 154 906–154 923, 2020.

[46] M. Piazza, M. Piccinini, S. Taddei, and F. Biral, "Mptree: A sampling-based vehicle motion planner for real-time obstacle avoidance," *IN PROCEEDINGS IFAC Proceedings Volumes*, 2024.

[47] K. Kreutz and J. Eggert, "Analysis of the generalized intelligent driver model (gidm) for merging situations," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 34–41.

[48] P. Cisek and J. F. Kalaska, "Neural mechanisms for interacting with a world full of action choices," *Annual review of neuroscience*, vol. 33, pp. 269–298, 2010.

[49] M. Da Lio, A. Cherubini, G. P. Rosati Papini, and A. Plebe, "Complex self-driving behaviors emerging from affordance competition in layered control architectures," *Cognitive Systems Research*, vol. 79, pp. 4–14, 2023. [Online]. Available: https://doi.org/10.1016/j.cogsys.2022.12.007

[50] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.62.1805

[51] Y. Li, L. Li, D. Ni, and Y. Zhang, "Comprehensive survival analysis of lane-changing duration," *Measurement*, vol. 182, p. 109707, 2021.

[52] P. Bosetti, M. Da Lio, and A. Saroldi, "On the human control of vehicles: an experimental study of acceleration," *European Transport Research Review*, vol. 6, no. 2, pp. 157–170, Jun. 2014.

[53] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.