European Research Council
Established by the European Commission

Self-assessment Oracles for Anticipatory Testing

TECHNICAL REPORT: TR-Precrime-2020-04

Gunel Jahangirova, Andrea Stocco, Paolo Tonella

Quality Metrics and Oracles for Autonomous Vehicles Testing

Project no.: 787703
Funding scheme: ERC-2017-ADG
Start date of the project: January 1, 2019
Duration: 60 months

Technical report num.: TR-Precrime-2020-04
Date: September, 2020
Organization: Università della Svizzera italiana
Authors: Gunel Jahangirova, Andrea Stocco, Paolo Tonella
Dissemination level: Public
Revision: 1.0

Università della Svizzera Italiana (USI)

Principal investigator: Prof. Paolo Tonella
E-mail: paolo.tonella@usi.ch
Address: Via Buffi, 13 – 6900 Lugano – Switzerland
Tel: +41 58 666 4848
Project website: https://www.pre-crime.eu/

Abstract

The race for deploying AI-enabled autonomous vehicles (AVs) on public roads is based on the promise that such self-driving cars will be as safe as or safer than human drivers. Numerous techniques have been proposed to test AVs, which however lack oracle definitions that account for the quality of driving, due to the lack of a commonly used set of metrics.
Towards filling this gap, we first performed a systematic analysis of the literature concerning the assessment of the quality of driving of human drivers and extracted 126 metrics. Then, we measured the correlation between such metrics and the human perception of driving quality when AVs are driving. Lastly, we performed a study based on mutation analysis to assess whether the 26 metrics that best capture the quality of AV driving according to the human study can be used as functional oracles. Our results, targeting the Udacity platform, indicate that our automated oracles can kill a high proportion of mutants at a zero or very low false alarm rate, and therefore can be used as effective functional oracles for the quality of driving of AVs.

Contents

1 Introduction

The competition for large scale deployment of autonomous vehicles (AVs) to consumers is premised on their potential to be as safe as or even safer than conventional vehicles driven by humans. As of today, AVs operate only in limited driving situations, and there is no consensus on how to evaluate the performance of existing in-development AVs. Proposals of guidelines to evaluate AVs exist [2, 21, 17], even though a consolidated regulatory framework for assessing the driving quality of AVs is still under investigation by government agencies, foundations, and private sector, not without obstacles.

The metrics that are most used by AV developers—miles driven and frequency of human intervention—are insufficient to demonstrate the safety of an AV. Such metrics are misleading and ubiquitously criticised because they do not give a fine-grained picture of the quality of driving. While for some sub-components of an AV, such as the traffic sign recognition system, we can use standard accuracy metrics, for the main driving functionalities, such as steering angle prediction from camera images, no fine-grained oracle exists and existing metrics (e.g., mean squared error) are only indirectly related to the quality of driving [20].

The research on the creation of accurate oracles for AVs is still in its infancy. Researchers have proposed solutions for testing AVs by means of test generation techniques [19, 18, 31, 41, 43, 1, 5, 6]. A number of papers use offline context-unaware thresholds as test oracles [19, 31, 41, 43] (e.g., the predicted steering angle should deviate from the ground truth angle by less than 5^0), which are immaterial to the specific driving task and therefore poorly indicative of the driving performance. Other works study DNN-enabled AVs within simulation platforms, but only considering violations to the safety requirements, i.e., collisions with other vehicles, pedestrians, or other objects, or episodes of off-road driving behaviours [18, 39, 36]. While being indeed vital requirements that must be satisfied, they represent only a *coarse-grained* proxy for the quality of driving.

In this paper, we propose a set of *fine-grained* driving quality metrics that are informed by a survey of the literature, are validated by means of a human study and are used to create oracles evaluated in online driving simulations. More specifically, we first performed an extensive study of the literature to determine what driving quality metrics have been used within experiments with human drivers related to different driving tasks. Second, we filtered the metrics that apply to the AV domain and measured their correlation with the quality of driving of AVs as assessed by the 63 participants involved in a human study. Finally, we have created driving quality oracles by jointly minimising the expected number of false alarms in good driving conditions and of missed alarms when the quality of driving drops. Such an optimisation problem was formulated in a form suitable for SMT solving [10].

We have evaluated the effectiveness of our oracles on the Udacity simulator for self-driving cars [42]. We have modified the simulator to log the proposed driving quality metrics (e.g., lateral deviation) automatically. Then, we created several different AV models by means of mutation testing. The results show that driving performance degradation of mutated models can be detected with small-sized oracles (as low as two to seven metrics combined). Particularly, mutant detection ranges between 73% and 100% when accepting a false alarm rate between 0% and 7%.

Our paper makes the following contributions:

Driving Quality Metrics A set of driving quality metrics that can be used to evaluate the performance of AVs.

Simulator An extension of the Udacity simulator to compute our metrics automatically during a simulation.

Evaluation An empirical validation of the metrics by means of a human study and of their usage as AV oracles.

Oracles A technique to automatically generate oracles that minimise false alarm and missed alarm rates.

2 Metric Extraction

2.1 Methodology

2.1.1 Database Search

While driving quality metrics for AVs is a relatively unexplored area, several studies assess the quality of *human* driving, possibly under the effect of some controlled treatment (e.g., when distracted by phone calls). Hence, we conducted a systematic search of such literature with the goal of identifying papers that assess the quality of human driving quantitatively, through metrics that can be potentially used to evaluate the autonomous driving capabilities of AVs.

We performed the literature search using Scopus, which is a comprehensive abstract and citation database hosting papers published in various peer-reviewed venues by multiple publishers (e.g., Elsevier, IEEE, ACM, Springer, Wiley). Scopus provides advanced search functionality and is one of the recommended scientific databases to conduct systematic studies in Software Engineering [26, 32].

Our search string combines terms related to the quality of driving with terms related to core objects involved in driving. We refined the search string in an iterative way to ensure that the maximum number of synonyms expressing both notions was covered. The final search query used with Scopus is:

```
((’driving quality’) OR (’driving performance’) OR (’driving assessment’) OR (’driving evaluation’))
AND
((’car’) OR (’vehicle’) OR (’road’) OR (’automobile’))
```

Application of this search string to the title, abstract, and keywords fields resulted in a list of 4,582 papers. We further filtered out the results by requiring that the list contains only conference or journal papers, and publications in the field of computer science or engineering. By applying this filter, our pool of papers was reduced to 2,277. After removing 7 duplicates, the final number of retained papers was 2,270.

2.1.2 Abstract Analysis

For each obtained paper, we had to ensure that it was related to the evaluation of driving quality. Since this task is difficult to automate, one of the authors conducted a manual analysis of all the 2,270 abstracts. Our inclusion criterion required that the abstract mentions that driving quality was quantified in some way. We excluded papers in which the vehicle used in the study is not a car, being, for example, agricultural machinery, two-wheeled inverted pendulum, or train. Moreover, we excluded papers if the adopted metrics pertain to the human perception system, such as driver’s heart rate, eye tracking, comfort, arm stiffness, posture information, or brain activity, or if the adopted metrics are related only to a very specific component of a car, such as the toe-in or the camber of the wheel. As a result of the manual evaluation of the abstracts, we identified 452 papers out of 2,270 (20%) relevant to our study.

Our goal was to extract a list of driving quality metrics from each of these papers. For 43 papers out of 452 (9%), the list was available directly from the abstract. For example, in the paper by Bartrim et al. [3], the abstract mentions: “*driving performance was assessed using a 30 min simulated driving task, with lateral (standard deviation of lane position [SDLP]; total number of line crossings [LC]) and longitudinal (standard deviation of speed [SDSP]) measures of vehicle control as outcome variables*”. In 257 papers (57%), the abstract did not provide a precise description of the metrics, yet it suggested that the paper could contain references to the use of some metrics. For example, in the paper by Caponecchia et al. [8], the abstract reports: “*results showed that participants reported feeling more*

drowsy in the afternoon, and performance impairments (increased lane deviations) were most evident in the morning". Finally, the remaining 152 papers (34%) did not contain any reference to the used metrics. Both of these last two cases would require manual analysis of the paper's full text, which is an expensive task. To keep our study feasible, we focused on the group of 257 papers whose abstract included some hook to quality metrics. Although the mentions of metrics in them were ambiguous, the chance of finding a precise definition in the full text was higher than for the group of 152 papers with no references whatsoever.

2.1.3 Full-Text Analysis

The authors then proceeded with downloading the full text of the 257 papers retained for the analysis. Scopus provides a plugin for one-click access to the corresponding PDF file. Unfortunately, for 31 papers the full text was not available through this functionality. We were able to collect the full text by contacting the authors for 12 of such papers; the remaining 19 papers were excluded from our study.

For each of the resulting 238 papers, the authors analysed the full text, identified the part that enumerates and defines the adopted quality metrics, extracted these metrics into a shared document, and provided precise definitions for custom, non-standard metrics. To ensure that all the authors understand the task and perform it consistently, we conducted a *pilot study* with 15 papers randomly chosen from the pool of 238 papers. In the pilot study, each of the three authors was assigned randomly to 10 papers, while ensuring that each paper has two assessors. For 10 papers out of 15 (66%), the authors had a full agreement on the list of extracted metrics. For 5 papers, the number of extracted metrics varied between the evaluators. All inconsistencies and conflicts were discussed and resolved during a consensus meeting. Moreover, the authors shared a document with the abbreviations and definitions of each extracted metric, to be used as a basis for the analysis of the remaining papers and avoid inserting duplicates.

For the final stage of the study, each author analysed 75 more papers. As the extraction process had been consolidated in the consensus meeting, each paper was assigned to only one evaluator. The shared file was updated during the task and all authors had access to it so that they could reuse the list of already extracted metrics, definitions, and abbreviations. Once all of the authors completed their task, another consensus meeting was conducted, to examine the final list of metrics, discuss all definitions and refine them to ensure a uniform and consistent use of abbreviations and operators.

2.2 List of Metrics

The authors of the analysed studies used either primitive metrics, typically visualised over time, or they applied some aggregate functions such as *min*, *max*, or *mean* in a time frame. During the metric extraction process, we both noted down the primitive metrics, along with the aggregation operators that were applied to them, if any. Across the surveyed 281 papers, 810 metric occurrences were found overall, from which we derived a list of 126 individual metrics. We also collected information about *events*, defined as either actions taken by the driver or the vehicle (e.g., braking), the consequences of these actions (e.g., collision), or changes in the driving environment (e.g., traffic sign occurrence). Some metrics are defined based on the occurrence of such events (e.g., counting them).

Table 1 provides information about all the extracted driving quality metrics. We found seven main categories: (1) generic metrics, (2) speed related metrics, (3) lateral position related metrics, (4) headway position related metrics, (5) steering related metrics, (6) braking related metrics, and (7) traffic sign-related metrics. For each driving quality indicator, the table reports whether it is a metric (m) or an event (e), its abbreviation, its name, the number of surveyed papers in which it was used as a primitive metric, and the operators with which it was combined, along with the

Table 1: List of Driving Quality Metrics, Events, and Operators

	Type	Abbr.	Metric Name	Uses	Operators
Generic					
	m	D	Distance	4	
	m	T	Time	1	
	e	CR	Crash	-	Count(14), TimeE(1)
Speed					
	m	SP	Speed	91	Mean(52), SD(42), DevT(7), Min(5), Var(5), TimeC(4), CountC(3), CountC(DevT)(2), Max(2), AbsDev(1), Diff(1), Mean(DevT)(1), SD(DevT)(1), Skew(1), TimeC(DevT)(1)
	m	ACC	Acceleration	29	Mean(6), Min(4), Max(5), SD(4), Count(1), CountC(1), Ent(1), Med(1), SoS(1), Var(1)
	m	AS	Acceleration Speed	0	Mean(1)
	m	TPP	Throttle Pedal Position	4	Mean(4), SD(3), CountC(2), Min(2), CountC(Max)(1), Ent(1), Max(1), Med(1), Mode(1), Var(1)
	m	TPS	Throttle Pedal Speed	1	PowF(1)
Lateral Position					
	m	LP	Lateral Position	61	SD(90), Mean(27), DevT(7), CountC(6), TimeC(4), RMSE(4), Var(3), Min(2), Max(2), AbsDev(1), CountC(DevT)(1), CV(1), SER(1), TimeC(DevT)(1)
	m	LS	Lateral Speed	3	Mean(1), SD(1)
	m	LA	Lateral Acceleration	4	Var(1)
	m	TO	Trajectory Offset	1	-
	e	LC	Lane Change	-	Count(12), TimeC(4), RA(2), TimeE(2), RT(1)
	e	LCR	Line Crossing	-	TimeE(4), Count(1)
Headway Position					
	m	HP	Headway Position	18	SD(6), Mean(3), Mean(DevT)(2), CV(1), Max(DevT)(1), SD(DevT)(1), SER(1)
	m	HT	Headway Time	7	Mean(3), SD(2), Min(1), TimeC(1)
	m	HS	Headway Speed	1	Min(1)
	m	DTP	Distance to Pedestrian	3	Min(1)
	m	DTV	Distance to Vehicle	2	-
	e	COL	Collision	-	Count(24), TimeC(5), TimeE(5), RT(1)
Steering					
	e	ST	Steering	-	RT(4)
	m	SA	Steering Angle	17	SD(19), Ent(8), Mean(6), Var(4), DevT(3), Max(3), CountC(2), RMSE(2), Med(1), Min(1), Mode(1), TimeC(1), PowF(1)
	m	SARR	SA Reversal Rate	17	-
	m	SAS	SA Speed	6	CountC(6), Max(1), Mean(1), Mode(1), SD(1), TimeC(1)
Braking					
	e	BR	Braking	-	RT(26), Count(4), TimeC(2)
	m	BPP	Brake Pedal Position	4	Mean(1), Max(1)
	m	BD	Braking Distance	3	-
Traffic Signs					
	e	TSO	Traffic Sign Occurrence	-	RT(3), RA(4)

number of papers mentioning each operator-metric combination (within brackets). For events, we do not report the number of primitive uses, as an event in isolation is not a performance indicator but rather it becomes one only when paired with an operator.

2.2.1 Operators

The list of operators that were applied to the metrics in the surveyed papers span from common arithmetic and statistical operators such as minimum (*Min*), maximum (*Max*), mean (*Mean*), median (*Med*), mode(*Mode*), sum (*Sum*), count (*Count*), standard deviation (*SD*), variance (*Var*), coefficient of variation (*CV*), root mean square error (*RMSE*), entropy (*Ent*), absolute mean deviation (*AbsDev*), sum of the squares (*SoS*), Pearson's first skewness coefficient (*Skew*), spectral power at a frequency band (PowF) and standard error of regression line (*SER*). Deviation from target (*DevT*) measures the absolute value of the difference between the target value of a metric and its actual value. *Diff* measures the difference between the maximum and minimum value of a metric, i.e., it characterises its range of values.

We also considered conditional operators, i.e., operators that aggregate only metric values that satisfy a given condition. *TimeC* denotes "time in condition", i.e., the period of time in which a given condition holds for a metric. Similarly, *CountC* stands for the number of times the condition holds for a metric. Both *TimeC* and *CountC* require a parameter in form of a condition. For example, *CountC(Braking)* will report how many braking actions were taken during the drive. Similarly, *TimeC(Speed* $>$ *60)* will compute the duration for which the value of speed was higher than 60. *TimeE* is used to express the time to an event, for example, time-to-collision, with an event being a required parameter of the operator. Reaction Time (*RT*) is calculated as $t' - t$, where t is the time of an event requiring an action, while t' is the time of the start of the action. Reaction Accuracy (*RA*) measures whether the correct reaction to an event has taken place. For example, if a driver has complied with 4 out of 5 traffic sign occurrence events during his trip, the reaction accuracy will be 0.8. All these three operators are applicable only to events.

2.2.2 Metrics

For the majority of metrics, such as speed or lateral position, there exist universal definitions that we have adopted. For paper-specific metrics that appeared in the analysed literature only once (e.g., trajectory offset), we used the definition in the corresponding paper. When the definitions for specific metrics varied across papers, or when the same definition was used for differently named metrics, we unified the variants and provide a precisely formulated definition. Each of the next paragraphs describes a family of metrics and provides the definitions that we have adopted for them.

Generic. The first block in Table 1 lists generic metrics such as the distance traveled by the vehicle, the time spent on the travel, and the number of crashes that took place.

Speed & Acceleration. Speed is the most often used metric in the literature. It is measured as the distance travelled by a vehicle in a unit of time. The operators applied to this metric most frequently are *Mean* and *SD*. Acceleration (*ACC*) is the rate of change of a car's velocity over time. Similarly, acceleration speed (*AS*) is the rate at which acceleration changes over time.

Throttle pedal regulates the vehicle's speed. Therefore, the Throttle Pedal Position (*TPP*) indicates the acceleration level the car might achieve, which can be used as a substitute for speed or acceleration. Throttle Pedal Speed (*TPS*) is defined as the rate at which *TPP* changes in a time unit. As Table 1 shows, *ACC* and *TPP* are widely used metrics that get combined with many different operators, while *AS* and *TPS* are custom metrics each used only once across all analysed papers.

Lateral Position. Lateral Position (*LP*) is defined as the distance between the center of the car and the center of the driving lane. The increase in the variation of *LP* is commonly associated with reduced lateral control. As Table 1 shows, this variation is most often measured using the *SD* operator.

Lateral Speed (*LS*) is measured as the change of the lateral position in a time unit. Similarly, Lateral Acceleration (*LA*) is the change of *LS* per unit of time. Trajectory Offset (*TO*) is a time-independent metric and is measured as the difference of the *LP* of the vehicle at the start and at the end of the driving task. As it can be seen in Table 1, *LP* as a primitive metric or combined with various operators is much more prevalent in the literature than *LS*, *LA*, or *TO*.

The events associated with the lateral position of a vehicle are Lane Change (*LC*) and Line Crossing (*LCR*). The *Lane Change Test* is a standard test where human drivers are asked to perform lane changes whenever prompted by road signs. Very often this experiment is conducted when drivers are distracted by a secondary task and their performance is evaluated against driving with no distractions. The quality of the lane changes, measured by the operators listed in the table, provides the metrics for comparison. *LCR* is defined as any vehicle's wheel touching any lane marking (except during lane changes). *LCR* is an indicator of poor human driving quality. The number of *LCR* occurrences per unit of distance (*Count* operator) or time to *LCR* (*TimeE* operator) are the main quantifiers of this event.

Headway Position. Headway Position (*HP*) is the distance to a lead vehicle, traveling in the ego vehicle's travel path. Headway Time (*HT*) is the time required for the ego vehicle to reach the lead vehicle. Headway Speed (*HS*) is measured as the change of *HP* over time. In contrast to *HP*, Distance to Vehicle (*DTV*) measures the distance to the closest vehicle in the back, left, or right of the ego vehicle. Distance to Pedestrian (*DTP*) is measured as a longitudinal margin to the pedestrian on the car's travel path. All these metrics are calculated under the assumption that both vehicles maintain their speed.

Collision (*COL*) with other vehicles or pedestrians is closely related to headway metrics. The overall number of collisions (*Count(COL*), and Time-To-Collision (*TimeE(COL)*) are the main operators used to characterise this event.

Steering. Steering is an essential part of the driving task and the changes in the performance of vehicles often manifest themselves as erratic steering patterns. Steering Angle (*SA*) is defined as the angle between the front of the vehicle and the steered wheel direction. The *SD* operator is the one most often applied to this metric. Steering Angle Reversal Rate (*SARR*) [28] is defined as the number, in a time interval, of steering angle reversals larger than a certain finite angle, or a gap. The magnitude of the gap is a key parameter of this metric. SA Speed (*SAS*) is the change of *SA* per unit of time.

Braking. Braking is activated to slow down or stop a moving vehicle. The reaction time to braking (*RT(BR)*) is the most widely used event-operator combination in our analysis. The Brake Pedal Position (*BPP*) in a vehicle indicates the intensity with which braking is being applied. Braking Distance (*BD*) is the distance a vehicle travels from the initial point of braking to the point in which the vehicles comes to a complete stop.

Conformance to Traffic Signs. Traffic Sign Occurrence (*TSO*) is an event that can trigger different actions in a vehicle, such as accelerating, braking, or changing the lane. A timely and correct reaction, measured with *RT* and *RA* operators, indicates conformance to the driving rules.

3 Empirical Validation of the Metrics

The *goal* of the empirical validation of the metrics that we extracted from the literature was to determine which of them characterise also the quality of AV driving. For such purpose, we took videos of AVs driving in a simulation environment and asked the participants in the study to assess the quality of driving apparent from the video using a 5-point Likert scale [33]. We then computed the correlation between the metrics measured in each road sector and the human assessment of the video of the same sector.

3.1 Simulation Platform and Measurement

3.1.1 Driving Simulator

We used the Udacity simulator for self-driving cars [42] as the main simulation environment. The simulator includes different track circuits and supports training and testing of AVs that performs behavioural cloning, i.e., the AV learns the lane-keeping functionality from a dataset of labeled driving scenes. We selected the Udacity simulator because it is a popular platform used by researchers to evaluate testing techniques for AVs [31, 41, 43, 39, 35].

The simulator provides three closed-loop tracks. *Lake Track (Track1)* is a road track designed with gentle bends, a wide road section, a bridge, and safety tires and road signs. The street is located on the edge of a lake, therefore the car may fall into the lake if lane-keeping is violated. *Jungle Track (Track2)* is a challenging road track with highly narrow curves. It is located within a mountain valley and is characterised by a smaller road section than Track1, and very limited visibility due to elevation changes. *Mountain Track (Track3)* is a mountain race track having one difficult and

narrow curve, long and gentle bends, and a wide three-lane road section.

3.1.2 Metrics

The Udacity simulator allowed us to record most of the quality metrics under investigation, many of which are readily available on a per-frame basis. *Speed* values are capped between 16 and 48 km/h. The *steering angle* predicted by the AV model ranges between -25 and +25 degrees. The *throttle* value is calculated using a linear interpolation between the minimum and maximum speed, ensuring the AV decreases the speed when the steering angle increase (e.g., in a bend).

Some metrics were not available in raw format, and thus we implemented them within the simulator. For example, we approximate *lane position* by means of the cross-track error (XTE), which is the distance from the center of the car's cruising position to the center of the road on the ideal trajectory between the planned route given by two consecutive waypoints [38]. To calculate the metrics that are a function of a unit of time–such as *Acc*, *LS*, *SAS*–we divided the overall duration of the simulation by the number of frames.[1]

The simulator used in this study is limited to only one AV, trained to follow the center of the road and stay within its initial lane. As such, we had to exclude metrics related to the presence of other vehicles or pedestrians (e.g., *HT*, or *DTV*), as well as those metrics that require a target value to be provided, such as *DevT*, *CountC*, *SARR*. Also, we excluded metrics that concern the presence of multiple lanes (e.g., *LC*), or traffic signs (i.e., *TSO*). Finally, we excluded metrics that were mentioned only once in the analysed papers, and metrics using *Var* operator since we consider the Standard Deviation operator *SD* operator, which is the square root of the variance, as a more standard and interpretable measure of data spread. In cases when these exclusion criteria led to the metric being left without any operators, we applied *SD* and *Mean* operators to it. Overall, 26 metrics were retained and used for the evaluation of the driving quality of AVs, which are listed in Table 2. The measurements used in this study were obtained when Nvidia's DAVE-2 [7] DNN model was driving. DAVE-2 consists of three 5x5 convolutional layers with stride 2 plus two 3x3 convolutional layers, followed by five fully-connected layers with dropout, and ReLu activation functions.

3.2 Research Questions

RQ_1 (Correlation). *Which metrics correlate with the human assessed quality of driving?*

RQ_1 aims to determine the correlation of the metrics used for assessing human driving quality with respect to the quality of autonomous driving, i.e., when the vehicle is driven by a computer program (i.e., a deep neural network in our study). We collect the human assessment of the videos of driving performed by a simulated AV model, calculate the values of the metrics for such driving simulation, and evaluate which metrics correlate with the human assigned scores.

RQ_2 (Independence). *Is there a minimal set of metrics that characterise the quality of AV driving adequately?*

RQ_2 aims to assess the correlation/independence of the retrieved metrics among each other, with the aim to identify a minimal set of metrics that is sufficient to characterise the quality of driving. Indeed, our driving quality metrics are *inter-related*. For instance, different operators applied to the same set of raw values (e.g., *Mean(Speed)*, and *Max(Speed)*) may give highly correlated results. In such cases, keeping only one of the two or more related metrics could be enough. We use Principal Component Analysis (PCA) to answer this question. In particular, we reduce the dimensionality of the metric space to its principal components and we determine the metrics that give the highest contribution to the principal components.

[1] The actual frame rate depends on the given hardware configuration. In our experiments, the frame rate was 15 fps.

Table 2: Correlation Results with Human Scores (RQ_1) and Independence (RQ_2). Bold values indicate statistical significance.

Metric	Correlation								Independence
	Lake Track		Jungle Track		Mountain Track		All Tracks		All Tracks
	cc	*p*	*cc*	*p*	*cc*	*p*	*cc*	*p*	VS
Std(Brake)	-0.44	**0.01**	-0.57	**0.01**	-0.17	0.29	-0.34	**0.00**	0.1917
Max(LP)	-0.70	**0.00**	-0.07	0.77	-0.64	**0.00**	-0.47	**0.00**	0.1887
Mean(Brake)	-0.40	**0.02**	-0.46	**0.03**	-0.17	0.29	-0.30	**0.00**	0.1868
Std(Speed)	0.22	0.21	-0.05	0.82	-0.41	**0.01**	-0.03	0.77	0.1805
Max(Acc)	0.32	0.07	-0.35	0.11	-0.42	**0.01**	-0.15	0.15	0.1773
Std(LP)	-0.60	**0.00**	-0.06	0.81	-0.37	**0.02**	0.38	**0.00**	0.1758
Min(Acc)	0.53	**0.00**	0.33	0.14	0.26	0.10	0.18	0.07	0.1744
Std(SA)	-0.67	**0.00**	-0.35	0.12	-0.22	0.15	-0.23	**0.02**	0.1744
Std(Acc)	-0.22	0.21	-0.39	0.08	-0.39	**0.01**	-0.24	**0.02**	0.1741
Min(Speed)	-0.12	0.51	0.29	0.20	0.39	**0.01**	0.13	0.21	0.1727
Max(SA)	-0.55	**0.00**	0.29	0.20	-0.24	0.12	-0.23	**0.03**	0.1724
Mean(TPP)	0.43	**0.01**	0.02	0.93	-0.29	0.06	-0.01	0.89	0.1706
Std(TPP)	-0.04	0.85	-0.39	0.08	-0.42	**0.01**	-0.19	0.07	0.1693
Std(LS)	-0.40	**0.02**	-0.35	0.12	0.07	0.67	-0.17	0.10	0.1650
Mean(Speed)	-0.19	0.29	0.23	0.31	0.32	**0.04**	0.08	0.42	0.1626
Mean(SAS)	0.60	**0.00**	0.33	0.15	-0.31	**0.04**	0.24	**0.02**	0.1540
Count(Braking)	-0.51	**0.00**	-0.01	0.96	-0.17	0.29	-0.06	0.59	0.1540
Std(SAS)	-0.32	0.07	-0.50	**0.02**	-0.32	**0.04**	-0.12	0.23	0.1520
Mean(SA)	-0.55	**0.00**	0.47	**0.03**	0.11	0.50	-0.03	0.74	0.1486
Mean(LS)	-0.44	**0.01**	-0.11	0.62	0.30	**0.05**	-0.19	0.07	0.1453
Mean(Acc)	0.64	**0.00**	-0.24	0.30	-0.23	0.15	0.28	**0.01**	0.1328
Mean(LP)	-0.75	**0.00**	-0.43	**0.05**	-0.69	**0.00**	-0.64	**0.00**	0.1293
Min(LP)	-0.24	0.17	-0.41	0.07	-0.45	**0.00**	-0.30	**0.00**	0.1268
Max(Speed)	0.23	0.20	0.19	0.40	-0.03	0.84	0.11	0.30	-
Count(Crash)	N/A	N/A	-0.55	**0.01**	-0.75	**0.00**	-0.46	**0.00**	-
Count(LCR)	-0.39	**0.03**	-0.56	**0.01**	-0.75	**0.00**	-0.54	**0.00**	-

3.3 Procedure and Metrics (RQ1)

Object. We chose DAVE-2 [7] as our AV model since it is a widely used model in DNN testing papers [31, 41, 43, 39]. More importantly, DAVE-2 exhibits realistic behaviours in simulated platforms, as well as realistic performance degradation when not appropriately trained [39], or when the model's architecture or training data get corrupted [24].

In order to obtain models with degraded driving performance, we created mutants of the original model by means of the DL mutation tool DeepMutation++ [22]. Specifically, we used DeepMutation++'s Gaussian Fuzzing mutation operator with ratio=0.03 for Lake Track and Jungle Track, and the Neuron Effect Block operator with ratio=0.03 for Mountain Track, to obtain representatives of poor driving quality models. Indeed, by visual inspection of the mutants driving in the simulation, we found them to exhibit visible misbehaviours with respect to the lane-keeping task even with no crashes.

For each available track, we executed DAVE-2 and the respective mutants on the Udacity simulator and recorded a screencast video. The simulation's video was split into smaller chunks such that (1) each chunk depicts the AV driving in one to a maximum of three consecutive track's sectors, and (2) each chunk's length ranges between $10 - 15$ seconds. These values were tuned by means of a pilot experiment, in which we found that they are appropriate to give the respondents enough information, while limiting the occurrence of multiple, possibly contrasting, candidate driving behaviours within the same video, which are difficult to rate with a single score. Shorter videos, on the other hand, were found useless due to lack of context.

Questionnaires. Overall, we prepared 12 questionnaires, four for each track, so as to cover the entirety of the tracks (for a total of 132 chunks). Each questionnaire contains between 11 video chunks, divided as follows: five videos were taken from the good driving quality model, five videos from the poor driving quality model, and one video of the poor driving quality model crashing/going off-road, which we inserted as attention check into the questionnaires (i.e., re-

spondents not giving the lowest score to this video were discarded, while the video itself was not used in later correlation analysis). Videos within the questionnaires were shuffled among respondents to mitigate carryover effects.

Subjects. We shared the 12 questionnaires within our personal contacts, trying to diversify the pool of potential respondents. We involved undergraduate and graduate students, PhD students and postdocs in computer science or other engineering fields. No participants were familiar with the purpose of the study. Possession of a driving license was the only requirement for selecting the participants, who were required to fill a pre-questionnaire including questions on driving experience and driving license.

Questionnaire Tasks. First, participants were given a short reference YouTube video of an AV driving in the Udacity simulator to familiarise themselves with the environment. No mention of the quality of driving was made at this stage. Participants were asked to assess the quality of driving of the 11 video chunks within each questionnaire. There was no time limit associated with the study. The quality of driving was assessed on a 5-point Likert scale [33]: (1) Very low; (2) Low; (3) Neither low nor good; (4) Good; (5) Excellent.

Metrics. We compute Pearson's correlation [25] between each metric and the Likert scale assessment made by questionnaire respondents. We also measure the correlation's p-value, which indicates the probability of lack of correlation. We consider metrics whose p-value is less than 0.05 where the sign of the correlation indicates whether they score good (positive correlation) or poor (negative correlation) quality of driving.

3.4 Procedure and Metrics (RQ2)

We used Principal Component Analysis (PCA), a method that reduces the dimensionality (i.e., number of features) of the vectors of a large dataset, while preserving as much variance (statistical information) as possible. We calculate the number of principal components required to preserve a total variance equal to 90% of the original variance (this is also called the *explained* variance). Then, for each such principal component, we get the list of metrics that contribute to it in the associated linear combination. The absolute value of each metric's coefficient in this combination indicates how much the metric contributed to the principal component. We calculate the variance score (VS) for a metric by multiplying its coefficient with the variance ratio explained by the corresponding principal component, and then summing up these values across all principal components. We then rank the metrics according to their variance score and select the top N for our further analysis. This ensures that we prioritise the metrics by their contribution to the principal components with the highest explained variance ratio.

3.5 Results

RQ_1 (Correlation). Overall, 63 participants answered our questionnaires. On average, we received 5 responses for each of the 12 questionnaires, with a minimum of three and maximum of 10 responses. To measure the *agreement rate* between the different raters, we used the Fleiss' κ [16] for evaluating the level of agreement between two or more raters on a categorical scale (Cohen's κ could not be used since we had occasionally more than two rates for the same questionnaire). Fleiss' κ takes a value between 0 and 1, where 0 denotes no agreement and 1 indicates perfect agreement [37]. For 11 questionnaires the agreement rate was statistically significant (i.e., p-value < 0.05). For nine out of 11 questionnaires the agreement rate was between 0.2 and 0.4, which is interpreted as a "fair" agreement according to Landis & Koch [27]. For the remaining two questionnaires the agreement rate was less than < 0.2, i.e., there was only a "slight" agreement, therefore we excluded these two questionnaires from further analysis.

As a result, we obtained human scores assigned to 96 different videos: 33 for Lake Track, 21 for Jungle Track, and 42 for Mountain Track. For each video, we calculated the values of the

driving quality metrics and the average score given by the human raters. Then, we assessed the correlation (or lack thereof) of each metric with the average human score using Pearson's correlation coefficient [25]. Table 2 (macro-column Correlation) lists all metrics and all correlation coefficients (*cc*) and p-values (p) both on a per-track and overall across all tracks.

The number of metrics that correlate with the human score in a statistically significant way is 16 for Lake Track, seven for Jungle Track, 15 for Mountain Track, and 13 considering all tracks. Overall, out of 26 metrics, 25 correlate with the human score (i.e., p-value $<= 0.05$) in at least one of the tracks. *Max(Speed)* is the only metric that does not correlate, which makes sense as the maximum speed is capped in the Udacity simulator. Our correlation results suggest that the extracted metrics are meaningful and quantify the quality of AV driving similarly to the human perception of driving.

Mean(LP) is the metric with the highest correlation coefficient for the overall dataset as well as for Lake Track and Mountain Track. This is expected, as the AV is trained to drive in the center of the lane, and deviations from such behaviour were rated negatively by the human raters in the questionnaires. In contrast, for the Jungle Track, the metric with the highest correlation coefficient is *Std(Brake)*. Given that this track is characterised by severe elevation changes, correct use of the braking by the AV is essential for successful completion of the driving task.

RQ_2 (Independence). We considered all 25 metrics that correlate with the human scores, but we excluded *Count(Crash)* and *Count(LCR)*, as the occurrence of these events is already evidence of the worst driving quality scenario. In fact, we are interested in metrics that are able to characterise a degraded quality of driving when such degradation does not necessarily lead to critical scenarios.

We applied PCA to our dataset, which is composed of 23 metrics and 96 different data points (vectors). PCA identified eight principal components that explain 90% of the variance. These components had different explained variance ratios, with the first component explaining 39% of the overall variance and the last one explaining only 2% of the overall variance.

Table 2 shows the VS values for all metrics, sorted by decreasing values of *VS*. Overall, the metric that contributes to explaining most variance is *Std(Brake)* (0.1917), followed by *Max(LP)* (0.1887), *Mean(Brake)* (0.1868) and *Std(Speed)* (0.1805). *Mean(LP)* and *Min(LP)* have the lowest VS scores, despite having the highest correlation score with the human raters. This can be explained by the nature of principal components: if two variables are linearly correlated, PCA will include only one of them among the top contributors to the principal components, since the other does not explain any further variance. However, by swapping them, so including the second and excluding the first, PCA might achieve a comparable explained variance ratio. This means that we cannot interpret the ordering of metrics in Table 2 to decide which metrics are good/bad indicators of the quality of driving. We can use such ordering only to select the top k metric that gives the highest, independent contribution to explain the variance, although alternative, equally good selections of k variables are possible.

4 Computation of Optimal Thresholds

Driving quality metrics can be used as oracles to assess the behaviour of an AV. Given a road sector x and a set of quality metrics $m_1(x), \ldots, m_n(x)$ collected in such sector, assuming that such metrics grow as the quality of driving degrades (a decreasing metric would just require to invert the inequality $\leq$ into $\geq$), we can define a quality of driving oracle as:

$$O(x) = m_1(x) \leq t_1 \wedge \ldots \wedge m_n(x) \leq t_n \tag{1}$$

The problem is how to determine optimal values of the thresholds $t_1, \ldots, t_n$, such that the oracle $O(x)$ evaluates to true in scenarios where the original AV model drives well, while it evaluates to false when it drives poorly. One way to obtain model instances that drive poorly is by mutation analysis: the ideal oracle should raise no false alarms in scenarios where the original model drives

Table 3: Example of metrics collected when D (resp. M_1) is driving in sector x_1 (resp. x_2)

	m_1	m_2	m_3	Killed
$D(x_1)$	**0.5**	0.3	0.4	-
$D(x_2)$	0.4	**0.6**	**0.9**	-
$M_1(x_1)$	**1.5**	0.3	**1.4**	Yes
$M_1(x_2)$	0.2	0.6	0.5	No

well (i.e., it should have only true negatives), while it should kill as many mutants as possible (i.e., it should maximise the number of true positives when mutants are driving) [23].

4.1 Problem Formulation

Given a model D that behaves correctly on a set X of driving sectors $x \in X$ and given a set of mutants $\{M_1, \ldots, M_j, \ldots\}$ of such a model, the metrics collected during actual driving of D or M_j are respectively indicated as $m_1^D(x), \ldots, m_n^D(x)$ and $m_1^{M_j}(x), \ldots, m_n^{M_j}(x)$. The *optimal thresholds* $t_1^*, \ldots, t_n^*$ are those that have only true negatives (no false alarms) when applied to the metrics collected by D and maximise the true positives (i.e., true alarms) when M_j is driving:

$$\forall x \in X, m_1^D(x) \leq t_1^* \wedge \ldots \wedge m_n^D(x) \leq t_n^* \tag{2}$$

$$t_1^*, \ldots, t_n^* = \arg\max_{t_1,\ldots,t_n} |\{x \in X | m_1^{M_j}(x) > t_1 \vee \ldots \vee m_n^{M_j}(x) > t_n\}| \tag{3}$$

For each metric m_i, the optimal threshold t_i^* can be easily shown to be:

$$t_i^* = \max_{x \in X} m_i^D(x) \tag{4}$$

In fact, this threshold ensures no false alarm because all values $m_i^D(x)$ are lower than or equal to their maximum value t_i^*. Moreover, since the metric value $m_i^{M_j}(x)$ triggers a true alarm if it is greater than the threshold, any threshold value greater than t_i^* can only have the same or less true positives when mutant M_j is driving. For a metric whose value decreases with a degraded quality of driving, $\max$ should be replaced by $\min$ in Equation (4).

Let us consider two driving sectors x_1, x_2 and three metrics m_1, m_2, m_3. Table 3 shows the values of the metrics collected when either the original model D or M_j was driving. The optimal thresholds ($t_1^* = 0.5$, $t_2^* = 0.6$, $t_3^* = 0.9$) ensure zero false alarms by construction when D is driving, and have two true positives and four false negatives when M_1 is driving (no metric is above the respective threshold when M_1 is driving in sector x_2; m_2 does not kill M_1 even in x_1). In this example, mutant M_1 is killed by metrics m_1 and m_3, since a true alarm is raised by these metrics in at least one sector (x_2), while metric m_2 is useless, since it does not kill M_1 in any sector. The oracles $O_1(x) = m_1(x) \leq 0.5$ and $O_2(x) = m_3(x) \leq 0.9$ are equally effective in mutation killing, in this small example.

We can generalise the previous problem formulation to allow a fraction ϵ of false positives when the original AV model D is driving:

$$\frac{|\{x \in X | m_1^D(x) > t_1 \vee \ldots \vee m_n^D(x) > t_n\}|}{|X|} \leq \epsilon \tag{5}$$

$$\max_{t_1,\ldots,t_n} |\{x \in X | m_1^{M_j}(x) > t_1 \vee \ldots \vee m_n^{M_j}(x) > t_n\}| \tag{6}$$

```
                          ;; true negatives (original)
                          ;; sector 1
                          (assert (<= (* s1 0.5) t1))
                          (assert (<= (* s1 0.3) t2))
;; thresholds             (assert (<= (* s1 0.4) t3))
(declare-const t1 Real)
(declare-const t2 Real)   ;; sector 2
(declare-const t3 Real)   (assert (<= (* s2 0.4) t1))
;; true negative sector   (assert (<= (* s2 0.6) t2))
(declare-const s1 Int)    (assert (<= (* s2 0.9) t3))
(declare-const s2 Int)
(assert (<= s1 1))        ;; true positives (mutant)
(assert (>= s1 0))        ;; sector 1
(assert (<= s2 1))        (assert-soft (or
(assert (>= s2 0))            (> 1.5 t1)
                              (> 0.3 t2)
;; tot true negatives = 1     (> 1.4 t3)))
(assert (= (+ s1 s2) 1)) ;; sector 2
                          (assert-soft (or
                              (> 0.2 t1)
                              (> 0.6 t2)
                              (> 0.5 t3)))
```

Figure 1: Optimal Threshold Constraints (SMT-LIB Language)

The case where no false positive is allowed is a particular case of the ϵ-formulation with $\epsilon = 0$. For the example in Table 3, if we allow $\epsilon = 0.5$ we could decrease t_1 from 0.5 to 0.4 (making x_1 a false positive), but this would not be beneficial since no additional mutant would be killed, or we could decrease both t_2 from 0.6 to 0.3 and t_3 from 0.9 to 0.4 (making x_2 a false positive). The latter is an optimal choice, since it increases the number of sectors where M_1 is killed from 1 to 2.

To solve the ϵ-formulation of the problem, we introduce binary variables $s_k \in \{0, 1\}$ which determine whether in nominal conditions (D is driving) there must be a true negative in sector x_k (then, $s_k = 1$) or a false positive is allowed ($s_k = 0$). The new equations that replace Equation (2) are:

$$\forall x_k \in X, s_k \cdot m_1^D(x_k) \leq t_1 \wedge \ldots \wedge s_k \cdot m_n^D(x_k) \leq t_n \tag{7}$$

$$\sum_{k=1}^{N} s_k = \lceil (1 - \epsilon) \cdot N \rceil, \quad with \;\; N = |X| \tag{8}$$

The constraints defined by Equations (7), (8) and (6) can be expressed in the SMT-LIB language as shown in Figure 1. The SMT solver shall satisfy all non retractable constraints (true negatives, on original) with a choice of s_i that respects the total number of required true negatives ($\lceil (1 - \epsilon) \cdot N \rceil$), and shall maximise the number of retractable (`assert-soft`) constraints (true positives, on mutant) that are satisfied (this is also called a *MaxSMT* problem).

4.2 Experimental Study

We assessed the possibility to find optimal oracle thresholds via SMT solving on the Udacity case study, with the aim of answering the following research question:

RQ_3 (Oracle). *Can we find metric thresholds that discriminate between the original model and mutants with no or few false alarms?*

We generated mutants for the driving models DAVE-2 [7] and Chauffeur [40] for the Lake Track, using two available DL mutation tools, *DeepMutation++* [22] and *DeepCrime* [11]. The purpose of the study is to assess whether computed oracles generalise across different AVs for the same lane-keeping task.

Table 4: Optimal Threshold Computation (RQ_3); FP = False Positives; FN = False Negatives

Size	FP Train (ϵ)	FP Test	FN Train	Mutants Killed
1	0 (0%)	0 (0%)		1 (3%)
1	1 (4%)	0 (0%)	23 (3%)	1 (3%)
1	2 (7%)	0 (0%)	23 (3%)	1 (3%)
1	3 (11%)	0 (0%)	23 (3%)	1 (3%)
2	0 (0%)	0 (0%)		2 (6%)
2	1 (4%)	0 (0%)	41 (5%)	2 (6%)
2	2 (7%)	0 (0%)	60 (8%)	16 (48%)
2	3 (11%)	2 (7%)	105 (13%)	33 (100%)
3	0 (0%)	0 (0%)		2 (6%)
3	1 (4%)	0 (0%)	41 (5%)	2 (6%)
3	2 (7%)	0 (0%)	60 (8%)	16 (48%)
3	3 (11%)	2 (7%)	105 (13%)	33 (100%)
4	0 (0%)	7 (25%)		33 (100%)
4	1 (4%)	0 (0%)	41 (5%)	2 (6%)
4	2 (7%)	0 (0%)	60 (8%)	16 (48%)
4	3 (11%)	2 (7%)	105 (13%)	33 (100%)
5	0 (0%)	10 (36%)		33 (100%)
5	1 (4%)	0 (0%)	42 (5%)	7 (21%)
5	2 (7%)	0 (0%)	61 (8%)	21 (64%)
5	3 (11%)	2 (7%)	106 (14%)	33 (100%)
6	0 (0%)	11 (39%)		33 (100%)
6	1 (4%)	0 (0%)	45 (6%)	18 (55%)
6	2 (7%)	2 (7%)	78 (10%)	33 (100%)
6	3 (11%)	12 (43%)	108 (14%)	33 (100%)
7	0 (0%)	11 (39%)		33 (100%)
7	1 (4%)	0 (0%)	75 (10%)	24 (73%)
7	2 (7%)	3 (11%)	99 (13%)	33 (100%)
7	3 (11%)	2 (7%)	129 (16%)	33 (100%)

DeepMutation++ proposes eight post-training mutation operators for feed-forward neural networks that are applicable to our case study. Five of these operators target weights and neurons of the model and have a *ratio* parameter that identifies what ratio of weights/neurons is affected by the mutation. We used three values of *ratio* equal to 0.01, 0.03, and 0.05 for each mutation operator [22]. The remaining three operators are applied to layers. To account for the dependence of these operators on the selected layer, we generated three different instances, each time selecting a different layer. Overall, we generated 24 mutants for each AV model.

Concerning *DeepCrime*, 15 out of 24 operators are applicable to our case study. Seven mutation operators have a parameter that can be selected from a range: the higher the value of the parameter, the more aggressive the mutation operator. For such operators, we applied a binary search over the range of each parameter, in order to exclude crashing mutants that are trivial to kill. The parameters of the remaining eight operators require a specific categorical value. For our experiments, we selected these parameter values randomly. Overall, we generated 26 mutations for each AV model.

We performed simulations with all 100 mutants (i.e., 50 for DAVE-2 and 50 for Chauffeur). We excluded 22 mutants of DAVE-2 and 17 of Chauffeur because they were experiencing either a crash or line crossing. For the remaining 61 mutants (28 for DAVE-2 and 33 for Chauffeur, respectively), we computed the values of the driving quality metrics. We then calculated the optimal oracle thresholds as described in Section 4.1 using DAVE-2 models as a reference (i.e., as a "training set" to learn the thresholds). Such oracles are able to kill all DAVE-2 mutants, and we evaluated their effectiveness on the Chauffeur models, which acted as our "test set". Specifically, the original Chauffeur model was used to assess the false positive rate, whereas its mutants were used to identify the fault detection capability of the oracle.

4.3 Results

RQ_3 (Oracle). The quality of the obtained oracles is a combination of their size, mutation killing capability, and number of false positives. Ideally, the perfect oracle will have a small size, no false positives, and perfect mutation killing score. Table 4 reports such values for different oracles. Column *Size* reports the number of variables considered in the oracle. For each oracle size n, we selected the top n metrics prioritised by the variance score (RQ_2). Column *FP Train* reports ϵ, the proportion of sectors in which we allowed FPs when constructing the oracle, using DAVE-2 to train the thresholds, while column *FP Test* reports the amount of FP witnessed during the evaluation of the oracle on Chauffeur. We also report the number of constraints in the Unsat core (column *FN Train*), which have been removed to find a solution to the threshold optimisation problem. Finally, Chauffeur's mutants are considered killed if a true positive alarm is raised in at least one of the sectors where the mutant drives.

We have analysed oracle sizes up to 10. However, since no improvements were observed after size seven, we limit the results to such a meaningful subset. Oracles containing only one variable have a very low mutation killing score. In contrast, when we use two variables, it is possible to obtain an oracle killing all the mutants with 7% FP rate on the test set. With more than two variables, no further reduction in the FP rate on the test set is observed while preserving the 100% mutation score. On the other hand, if we aim for 0% FP on the test set, but we also accept a lower mutation score, we can identify a couple of interesting configurations: 48% mutation score and 0% FP on test set when using two variables, 64% mutation score and 0% FP on test set when using five variables, and 73% mutation score with 0% FP with seven variables.

Overall, we can notice that with a number of variables between two and five we have several solutions with interesting trade-offs between FP rate and mutation score, ranging from no false positive but reduced mutation score, to maximum mutation score with a few residual false alarms.

4.4 Threats to Validity

Internal Validity. The main threat to internal validity concerns our custom implementation of metrics within the simulator. We tested such implementation extensively. Another possible threat is the training of our own AVs and mutants, which may exhibit a large number of misbehaviours if trained inadequately. We mitigated this threat by training and fine-tuning the driving models using the guidelines from existing papers [39] and by using state-of-the-art DL mutation testing tools.

External. Our results are as representative as the Udacity simulator's capabilities to reflect the AV's model in the real-world. While there exist datasets of real driving scenes used to train robust AV models, these datasets cannot be used in our study because we cannot compute our metrics for such driving scenarios. We used a limited number of AVs and tracks in our evaluation, which also pose a threat in terms of generalisability of our results. We tried to mitigate this threat by choosing real-world AV models which achieved competitive scores in the Udacity challenge competition.

Reproducibility. All our results and references, the simulator, and subjects are available in our replication package [34].

5 Related work

Simulation Platforms. The Udacity simulation platform for the behavioural cloning task [42] and BeamNG [4] have been used in multiple papers on testing AVs [18, 39, 36]. Both platforms, however, to the best of our knowledge, do not include driving quality metrics to assess AV models.

CARLA [12] reports safety metrics such as off-road episodes and collisions with pedestrian, or other vehicles [14]. Apollo provides 12 metrics that include both safety violations, and other vio-

lations such as traffic light or lane changes [13]. DeepDrive scores only a test scenario consisting of a car proceeding on a four-way junction, scored with a combination of total trip time and g-force caused by acceleration [15].

Differently, the driving quality metrics proposed in this paper were retrieved by a comprehensive analysis of the literature. Being more generic, we expect them to be largely applicable to a wide variety of driving scenarios and simulators. In this work, we implemented and evaluated them within the Udacity simulator, providing an enriched platform to researchers that use this simulator in their studies.

Empirical Studies. Codevilla et al. [9] investigate the relation between online and offline metrics for AVs. Offline prediction errors are not necessarily correlated with driving quality, and two models with comparable error prediction rates may differ substantially in their driving quality. Similarly to Codevilla et al. [9], Haq et al. [20] performed an empirical study comparing offline and online testing of DNNs for AVs. The goal was to understand whether simulator-generated data can be a reliable proxy for real-world data. Specifically, all severe violations exposed by simulations are also exposed by offline techniques/measures, but the opposite is not true. Compared to our study, the metrics used in both empirical studies [9, 20] are coarse-grained driving quality indicators.

Oracles in Test Generation. In the *offline* setting, several approaches generate input images that trigger steering angle inconsistencies between multiple autonomous driving systems [31], or between the original and transformed driving driving scenarios [41, 43, 30]. The oracle used in such works consists of measuring whether the deviation between actual and reference steering angles is higher than a given threshold.

Concerning *online testing* techniques for AVs, researchers proposed search-based techniques to generate scenarios that cause AVs to misbehave [18, 39, 36, 29]. All such papers consider coarse-grained functional oracles checking only the violation of safety requirements, such as collisions or off-road episodes. Differently, in our work, we introduce fine-grained metrics that discriminate the actual quality of driving even in the absence of such extreme misbehaviours, which in our framework represent the simplest and most straightforward, but also least discriminative, form of oracle for AVs.

6 Conclusions and future work

Evaluating the driving performance of autonomous driving systems requires testing techniques equipped with effective functional oracles able to discriminate good from bad quality of driving. Towards this aim, in this paper we extracted a large set of metrics regarding the quality of human driving from the literature, and studied their applicability to the autonomous driving domain. The results of our user study suggest that our list of metrics do correlate with human perceived quality of driving. Furthermore, we selected a minimal set of metrics to automatically construct a driving quality oracle that was able to discriminate between robust and weak AV models (obtained by means of mutation analysis) with few to no false alarms.

As part of our ongoing and future work, we plan to include more subjects (and corresponding driving tasks) into our study, and analyse the sensitivity, reliability and applicability of the proposed metrics in different driving environments. Furthermore, we are planning to develop an abstraction layer to allow easier extensibility of our infrastructure to other driving simulators and testing frameworks.

References

[1] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas 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*, ASE 2018, pages 143–154, New York, NY, USA, 2018. ACM.

[2] AIDE - Adaptive Integrated Driver-vehicle InterfacE. `http://www.aide-eu.org/pdf/sp2_deliv_new/aide_d2_2_5.pdf`, 2005.

[3] Karly Bartrim, Brianna McCarthy, Danielle McCartney, Gary Grant, Ben Desbrow, and Christopher Irwin. Three consecutive nights of sleep loss: effects of morning caffeine consumption on subjective sleepiness/alertness, reaction time and simulated driving performance. *Transportation research part F: traffic psychology and behaviour*, 70:124–134, 2020.

[4] BeamNG GmbH. BeamNG.research. `https://beamng.gmbh/research/`, 2018. Online; accessed 18/08/2019.

[5] 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)*, pages 63–74, September 2016.

[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)*, pages 1016–1026, May 2018.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[8] Carlo Caponecchia and Ann Williamson. Drowsiness and driving performance on commuter trips. *Journal of safety research*, 66, 2018.

[9] Felipe Codevilla, Antonio M. López, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. *CoRR*, abs/1809.04843, 2018.

[10] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[11] DeepCrime. `https://github.com/deepcrime-tool/DeepCrime`, 2020.

[12] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017.

[13] Apollo's Dreamland. `https://github.com/ApolloAuto/apollo/blob/bf2b3730835fd5a98f91a5fb00f9d82ef750be9d/docs/specs/Dreamland_introduction.md`, 2019.

[14] CARLA's Driving Benchmark Performance Metrics. `https://carla.readthedocs.io/en/0.8.4/benchmark_metrics/`, 2019.

[15] DeepDrive Voyage. `https://deepdrive.voyage.auto/leaderboard/`, 2019.

[16] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[17] Laura Fraade-Blanar, Marjory S. Blumenthal, James M. Anderson, and Nidhi Kalra. Measuring Automated Vehicle Safety - Forging a Framework. Technical report, Rand, 2018.

[18] Alessio Gambi, Marc Mueller, and Gordon 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*, ISSTA 2019, pages 318–328, New York, NY, USA, 2019. ACM.

[19] Divya Gopinath, Guy Katz, Corina S. Păsăreanu, and Clark Barrett. Deepsafe: A data-driven approach for assessing robustness of neural networks. In Shuvendu K. Lahiri and Chao Wang, editors, *Automated Technology for Verification and Analysis*, Cham, 2018. Springer International Publishing.

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

[21] HASTE - Human Machine Interaction and the Safety of Traffic in Europe. `https://trimis.ec.europa.eu/project/human-machine-interaction-and-safety-traffic-europe#tab-outline`, 2005.

[22] Qiang Hu, Lei Ma, Xiaofei Xie, Bing Yu, Yang Liu, and Jianjun Zhao. Deepmutation++: A mutation testing framework for deep learning systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1158–1161. IEEE, 2019.

[23] Gunel Jahangirova, David Clark, Mark Harman, and Paolo Tonella. Test oracle assessment and improvement. In *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA)*, pages 247–258, 2016.

[24] Gunel Jahangirova and Paolo Tonella. An empirical evaluation of mutation operators for deep learning systems. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 74–84. IEEE, 2020.

[25] Wilhelm Kirch, editor. *Pearson's Correlation Coefficient*, pages 1090–1091. Springer Netherlands, Dordrecht, 2008.

[26] B. Kitchenham and S Charters. Guidelines for performing systematic literature reviews in software engineering, 2007.

[27] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174, 1977.

[28] John R. McLean and Errol R. Hoffmann. Analysis of drivers' control movements. *Human Factors*, 13(5):407–418, 1971.

[29] Galen E. Mullins, Paul G. Stankiewicz, R. Chad Hawthorne, and Satyandra K. Gupta. Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. *Journal of Systems and Software*, 137:197–215, 2018.

[30] S. Müller, D. Hospach, O. Bringmann, J. Gerlach, and W. Rosenstiel. Robustness evaluation and improvement for vision-based advanced driver assistance systems. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2659–2664, September 2015.

[31] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 1–18, New York, NY, USA, 2017. ACM.

[32] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18, 2015.

[33] Victor R. Preedy and Ronald R. Watson, editors. *5-Point Likert Scale*, pages 4288–4288. Springer New York, New York, NY, 2010.

[34] Replication Package. `https://github.com/guneljahan/driving_quality_metrics/`, 2020.

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

[36] Vincenzo Riccio and Paolo 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*, ESEC/FSE '20, 2020.

[37] Julilus Sim and Chris C. Wright. The kappa statistic in reliability studies: Use, interpretation, and sample size requirements. *Physical Therapy*, 2005.

[38] Andrea Stocco and Paolo Tonella. Towards anomaly detectors that learn continuously. In *Proceedings of 31st International Symposium on Software Reliability Engineering Workshops*, ISSREW 2020. IEEE, 2020.

[39] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. Misbehaviour prediction for autonomous driving systems. In *Proceedings of 42nd International Conference on Software Engineering*, ICSE '20, page 12 pages. ACM, 2020.

[40] Team Chauffeur. Steering angle model: Chauffeur. `https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur`, 2016. Online; accessed 18/08/2019.

[41] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 303–314, New York, NY, USA, 2018. ACM.

[42] Udacity. A self-driving car simulator built with Unity. `https://github.com/udacity/self-driving-car-sim`, 2017. Online; accessed 18/08/2019.

[43] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz 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*, ASE 2018, pages 132–142, New York, NY, USA, 2018. ACM.