Machine Learning (2020) 109:1103–1126
https://doi.org/10.1007/s10994-020-05872-w

Engineering problems in machine learning systems

Hiroshi Kuwajima[1] · Hirotoshi Yasuoka[2] · Toshihiro Nakae[2]

Received: 1 March 2019 / Revised: 1 July 2019 / Accepted: 13 February 2020 / Published online: 23 April 2020

Abstract

Fatal accidents are a major issue hindering the wide acceptance of safety-critical systems that employ machine learning and deep learning models, such as automated driving vehicles. In order to use machine learning in a safety-critical system, it is necessary to demonstrate the safety and security of the system through engineering processes. However, thus far, no such widely accepted engineering concepts or frameworks have been established for these systems. The key to using a machine learning model in a deductively engineered system is decomposing the data-driven training of machine learning models into requirement, design, and verification, particularly for machine learning models used in safety-critical systems. Simultaneously, open problems and relevant technical fields are not organized in a manner that enables researchers to select a theme and work on it. In this study, we identify, classify, and explore the open problems in engineering (safety-critical) machine learning systems—that is, in terms of requirement, design, and verification of machine learning models and systems—as well as discuss related works and research directions, using automated driving vehicles as an example. Our results show that machine learning models are characterized by a lack of requirements specification, lack of design specification, lack of interpretability, and lack of robustness. We also perform a gap analysis on a conventional system quality standard SQuaRE with the characteristics of machine learning models to study quality models for machine learning systems. We find that a lack of requirements specification and lack of robustness have the greatest impact on conventional quality models.

Keywords Machine learning · Software engineering · Systems engineering · Safety critical systems · Automated driving · Quality models

Editors: Mariam Kiran and Samir Khan.

✉ Hiroshi Kuwajima
hiroshi.kuwajima.j7d@jp.denso.com; kuwajima@ok.sc.e.titech.ac.jp

Hirotoshi Yasuoka
hirotoshi.yasuoka.j2z@jp.denso.com

Toshihiro Nakae
toshihiro.nakae.j8z@jp.denso.com

1 DENSO CORPORATION & Tokyo Institute of Technology, Tokyo, Japan

2 DENSO CORPORATION, Tokyo, Japan

1 Introduction

Recent developments in machine learning techniques, such as deep neural networks (NNs), have led to the widespread application of systems that assign advanced environmental perception and decision-making to computer logics learned from big data instead of manually built rule-based logics (Bird et al. 2017). Highly complex machine learning techniques such as NNs have been studied for decades, however until recently, we have suffered from numerous training data to train complex models properly and computing methods to perform high computational complexity of training such models. The availability of big data and affordable high-performance computing, such as deep learning frameworks on off-the-shelf graphics processing units (GPUs) (Jia et al. 2014), have made highly complex machine learning techniques practical for various applications including automatic speech recognition (Graves et al. 2013), image recognition (Krizhevsky 2012), natural language processing (Sutskever 2014), drag discovery (Vamathevan et al. 2019), and recommendation systems (Elkahky et al. 2015).

Machine learning models are becoming indispensable components even of systems that require safety-critical environmental perception and decision-making such as automated-driving systems (Li et al. 2017). A safety-critical systems is a system whose failure may result in safety issues such as death or serious injury to people. For safety-critical systems, worst-case performance is more important than average performance, and developers are held strictly accountable. However, for human society to accept such safety-critical machine learning systems, it is important to develop common engineering frameworks, such as quality measures and standard engineering processes, to manage the risks of using machine learning models and systems that include machine learning models (Koopman and Wagner 2016). Such frameworks, and ultimately the quality assurance based on them, have an impact on social receptivity because they can be one of the approaches used to deliver safety and security. In fact, recent accidents caused during the use of several experimental automated vehicles have revealed the imperative need to address the upcoming social issue of (quality) assurance based on such frameworks (https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/autonomousveh_ol316+). Engineering frameworks such as standard development processes have been studied for conventional systems and software for years, and machine learning systems also need such frameworks that engineers can follow. In order to establish engineering frameworks, it is necessary to visualize and organize these open problems; thus, experts from numerous different technical fields discuss these problems in depth and develop solutions driven by engineering needs.

In this study, we review the open engineering problems associated with safety-critical machine learning systems and also present related works and future directions for research. We hypothesize an ideal training process that connects deductive requirements and data-driven training by considering test data as a requirements specification and training data as a design specification; thereafter, we review open problems for the process. Our results show that machine learning models are characterized by a lack of requirements specification, lack of design specification, lack of interpretability, and lack of robustness. In addition, we discover that requirements specification and verification for open environments are key aspects of machine learning systems. We also study quality models for machine learning systems, which can be used for future requirements and evaluations of these machine learning systems. Our results show that a lack of requirements specification and lack of robustness have the greatest impact on conventional system quality models.

2 Background

An automated driving vehicle is a vehicle that operates without human input. Automated driving has not been built as a stand-alone system in a vehicle but can be realized using a system comprising clouds, roadside devices (fog or cloud edge), and automated driving vehicles (edge) (Borraz et al. 2018), which create and update high-precision digital maps (Poggenhans et al. 2018) while cooperating with peripheral vehicles. An in-vehicle automated driving system installed in a vehicle comprises multiple subsystems for perception, planning, and control; such a system realizes automated driving operations in cooperation with clouds and roadside units (Ali and Chan 2011). For simplicity, in this paper, we focus on these in-vehicle automated driving systems. Each perception, planning, and control subsystem may contain necessary machine learning models. Supervised learning models (Krizhevsky 2012) and reinforcement learning models (Mnih et al. 2013) can be used for perception and planning, while non-machine learning control algorithms can be used for control. In order to build a machine learning system, it is necessary to define its engineering processes and quality measures in advance, then follow and measure them strictly during development time. Conventional systems were developed in a rigorous development process involving requirement, design, and verification, cf. V-Model (INCOSE 2015) (a graphical representation of a systems development lifecycle).

In this study, we identify open engineering problems at two levels—systems and machine learning models—and use an automated driving system as an example of a safety-critical machine learning system. We proceed to investigate the problems in terms of the three steps of the development process: requirement, design, and verification. The two levels and three steps are illustrated in Fig. 1. Notably, many of the problems considered in

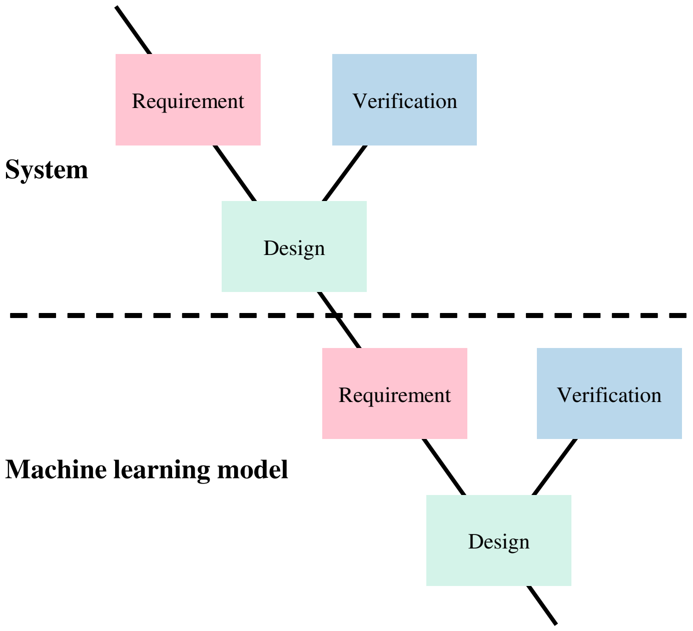


Fig. 1 Engineering process of machine learning systems

this paper do not occur only in automated driving systems but also generally in safety-critical systems.

This study is related to preceding studies (Salay 2017; Falcini and Lami 2017) that studied the applicability of ISO 26262 (ISO 26262 2018) and Automotive SPICE (VDA 2015) to automotive software using machine learning and deep learning. Our work assumes a more general development process to show open problems; we examined quality models for machine learning systems, based on a conventional system and software quality standard, Systems and software Quality Requirements and Evaluation (SQuaRE) (ISO 2014), which has not been done in previous studies.

3 Engineering machine learning models

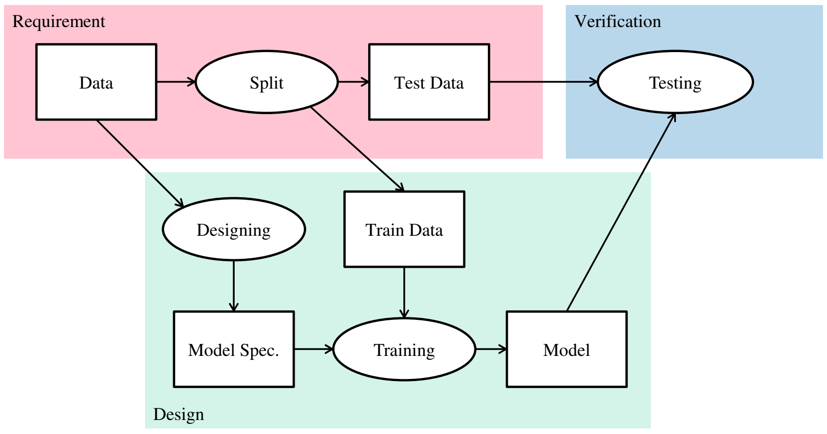
A machine learning model is acquired by executing a training algorithm with a model structure and training data sets for inputs, while trained models are evaluated using test datasets (Murphy 2013). This is a data-driven inductive method that differs from the deductive development used for conventional systems. In this paper, we call a machine learning model that has undefined parameters a "model structure." In order to use machine learning models in a deductively engineered system, it is necessary to break down the data-driven training of model parameters into requirements, designs, and verifications, particularly for models used in safety-critical systems.

We hypothesize the engineering process for machine learning models in Fig. 2. The dotted boxes in the figure illustrate the differences between the conventional training process and hypothesized training process. A requirement of machine learning models can be the specification of test data, although the current practice is to divide the original data into training and test data sets (Stone 1974; Arlot et al. 2010). The design process then specifies or builds the training data to achieve high performance in the test data, with the model requirements as a background. The explicit specification of test data and training data addresses a lack of requirements specification and a lack of design specification, respectively. In the current practice, the verification of machine learning models is measured using performance metrics on the test data. However, we consider it important to check properties that cannot be measured using the test data, such as robustness and interpretability.

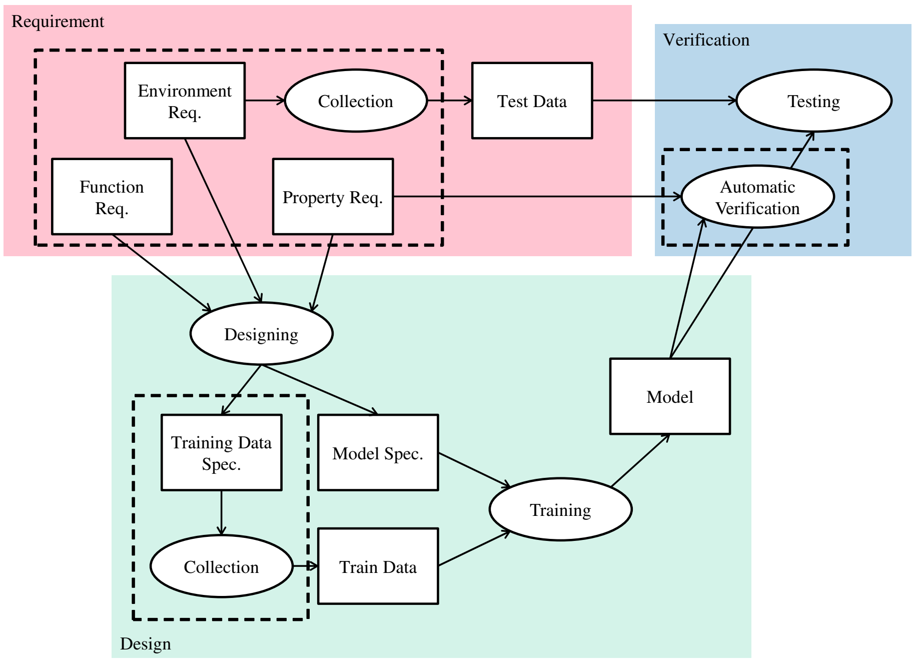
In the following subsections, we introduce our ideas related to the requirements, designs, and verifications of machine learning models, as well as research directions and related works.

3.1 Requirements of machine learning models

Most current machine learning research undoubtedly assumes that test data is given (Krizhevsky 2012; He et al. 2016); it is the main part of a model's requirements. Test data must be carefully specified at the beginning of development, by either the developers or contractees of the machine learning model, and must be agreed upon by their contractors. Thus, the main open engineering problem here is the deductive definition of the requirements for machine learning models and their test data to enable the test data to connect with deductive requirements and data-driven training. In machine learning, the roles of training data and test data must be considered to be different. While training data is used to improve the performance of a machine learning model (James et al. 2014), we propose considering



(a) Conventional training process



(b) Hypothesized training process

Fig. 2 Engineering process of machine learning models

the test data to accurately reflect the environmental conditions in operation. However, in practice (Ben-David et al. 1997), when all data obtained at the time of development are divided, some are used as training data and the others become test data (Stone 1974; Arlot et al. 2010). For simplicity, we ignore validation data for model selection. Despite the ultimate goal of machine learning models to work well in operation, we test machine learning models on test data, which originates from the same source as training data. In this manner, the training and test data are approximately equally distributed, but their relationship to the operational data (which is the actual target of the model) is unknown or it is implicitly assumed that the training, test, and operational data sets are similar (Bickel et al. 2009). In other words, machine learning models are trained using data-driven methods that lack requirements specification.

In particular, in a safety-critical machine learning system, it is necessary to specify the distribution of test data (considering the operational environment the system will actually be operated in) and to collect the test data based on these specifications. By accepting an a priori viewpoint of the distribution of test data, we can define the assumed environment deductively and collect data inductively. Moreover, by assuming the distribution of test data, we can discuss the operational domain (operational data distribution) for requirements specification. Operational data tend to change with time, thereby deteriorating model performance in operation (Tsymbal 2004; Webb et al. 2016). The deviation between test data used during development and operational data can become larger with time from what it was when development was completed. This phenomenon is referred to either as covariance shift (Bickel et al. 2009), distributional shift (Amodei et al. 2016), or concept drift (Tsymbal 2004). If the operational data trend changes from that of test data, then machine learning models trained on the test data do not work on the changed operational data. Thus, it is important to check for consistency between operational data and test data (assuming the original environment) and to either make the machine learning models follow the operational data in a continuous maintenance process or to, at least, detect the deviation between test and operational data. A lack of requirements specification is a barrier to this.

3.1.1 Related works and research directions for requirements of machine learning models

Although it does not incorporate the specification of test data, *i.e.*, requirements specification, runtime monitoring of neuron activation patterns is an approach to detect change points (Cheng et al. 2018). It creates a monitor of neuron activation patterns after training time, and runs the monitor at operation time to measure the deviation from training time. Change is detected when the activation pattern at operation time becomes detached from the neuron activation pattern at training time. Neuron activation patterns on test data may implicitly include the model requirements as a background.

Even in the current development of in-vehicle automated driving systems, the test data would be collected assuming the operational environment, in order to make the distribution of the operational data and that of the test data as consistent as possible. However, the methods used to describe the assumed environment of machine learning models are not organized. In particular, specific methods are required to define the completeness of test data. In previous literature, Computer Vision-Hazard and Operability Studies (Zendel et al. 2015) defined a catalogue of challenges (risks) for computer vision algorithms. The catalogue has 1469 manually registered risks as of now. Including all CV-HAZOP risks can be a test data coverage in computer vision problems. When systematically testing machine learning models to achieve test data coverage, we experience combinatorial explosion while guiding the data sampling process. In previous literature, quantitative projection coverage was used to resolve such combinatorial explosion (Cheng et al. 2018).

Although these previous works focused on combinatory environments, the importance or criticality of each environment could change. For example, criticality of misclassification of pedestrians may be high in daytime city street, whereas that of vehicles may be high in night time highway. CV-HAZOP proposes that the catalogue of challenges creates a basis for referencing criticalities for each risk and calculating criticality coverage (Zendel et al. 2015). Figure 3 illustrates our proposed example of requirements specification. Test data must have attributes, such as time and weather, and their distributions that are based on the assumed environment (Fig. 3a). Recent public driving data sets have such attributes. For example, BDD100K (Yu et al. 2018) has weather conditions, including sunny, overcast, and rainy, different times of day, including daytime and nighttime, as well as scenes, including city street, gas stations, highway, parking lot, residential, and tunnel. Further, since the required performance may change for each environment, it is necessary to express the association between the assumed environment and the required performance. Each condition of the test data distribution can have a different confusion matrix (or other performance metrics) that machine learning models will have as desired values (Fig. 3c).

3.2 Design of machine learning models

A machine learning model is automatically obtained by training the parameters of a model structure using training data. Thus, specifications cannot be designed a priori-that is, machine learning models lack design specifications. This limitation is essential and unavoidable because high-performance machine learning models are developed by learning high-dimensional parameters from data that engineers cannot manually specify. However, in the development of a safety-critical machine learning system, it is necessary to record the model structure, training data, and training system, including training specifications-such as hyper parameters, initial parameters, learning rates, and random number seeds-to secure the reproducibility of the training process.

Engineers cannot design the training; however, they can design the training data. Training data, as a large indirect part of the design specification, coupled with training

		time	
		day	night
weather	fine	40%	30%
	rainy	20%	10%

(a) Environment specification

		predicted	
		pedestrian	vehicle
actual	pedestrian	90%	10%
	vehicle	20%	80%

(b) Performance specification for fine × day †

		predicted	
		pedestrian	vehicle
actual	pedestrian	85%	15%
	vehicle	15%	85%

(c) Performance specification for fine × night

		predicted	
		pedestrian	vehicle
actual	pedestrian	90%	10%
	vehicle	20%	80%

(d) Performance specification for rainy × day

		predicted	
		pedestrian	vehicle
actual	pedestrian	85%	15%
	vehicle	15%	85%

(e) Performance specification for rainy× night ‡

† E.g., there are many pedestrians in fine daytime, and they are prioritized.
‡ E.g., there are many vehicles in rainy nights, and they are prioritized.

Fig. 3 Example environment requirements specification (data distribution matrix) and performance requirements specification (confusion matrix)

specifications is carefully designed to achieve the requirements specification. In this manner, the lack of design specification is indirectly remedied. Yet, to the best of our knowledge, there is no standard or widely accepted process of designing training data for machine learning models.

3.2.1 Related works and research directions for design of machine learning models

One of the challenges with the lack of design specification is the establishment of a training process for machine learning models by designing training data and models. Training data must be designed in the process by iteratively identifying the weak points of the model and then generating or collecting additional data for training. A previous suggestion (Ng 2015) indicates that a criteria for growing training data is that the training error is low while the test error is high; however, the suggestion does not show what types of data must be added. It is known that deep learning models, in particular, easily fit a random labeling of the training data (Zhang et al. 2016) and, thus, the distribution of training data is important.

3.3 Verification of machine learning models

Machine learning models are mainly verified by running a model on test data; however, certain properties of a machine learning model, such as robustness, cannot be evaluated with test data. Therefore, we introduce property checking in the verification of machine learning models.

An increasing stability against disturbance, or a lack of robustness, is key to the verification of machine learning models. It has been reported that image recognition models incorrectly recognize slight noise that cannot be recognized by humans with high confidence, thereby creating what are called adversarial examples (AEs) (Szegedy et al. 2013). An AE is known to have model-independent versatility and is an issue that can threaten the safety of automated driving systems, depending on image recognition. For example, when evaluating robustness against an AE as fault tolerance, it is necessary to artificially generate perturbations around data points. We can generate an AE close to a data point specified in the requirements and quantify the robustness using the maximum radius in which the model can yield correct answers.

The inference processes of advanced machine learning models—such as NNs—are considered black boxes, and machine learning models lack interpretability. In this context, a black box refers to a situation where, although feature activations can be observed physically, the actual phenomenon cannot be understood. That being said, safety-critical systems must exhibit interpretability and transparency. The interpretability of machine learning models has been well-researched recently and there are several methods for addressing it. LIME (Ribeiro et al. 2016) is one of the most well-known methods for improving interpretability. It derives a simple interpretable model to explain the behavior of an original model around a given data point. NN visualization (Grün et al. 2016) also shows great promise to improve interpretability. Object detectors emerging in deep scene CNNs is an NN visualization that intentionally performs occlusion on input data and specifies the region where the inference result changes drastically as a region of interest (Zhou et al. 2015; Zhou and Khosla et al. 2016); another method back-propagates activation values from the influencer nodes during the subsequent feature extraction process to identify the region of interest (Zeiler and Fergus 2014) and generate heat maps (Shrikumar et al. 2016; Binder et al. 2016; Montavon et al. 2017; Simonyan et al. 2014) for convolutional NNs. Further, interpretability is also useful for performance improvement, debugging during training, and validating of training results. Developers can understand the internal behavior of a trained NN to train higher performance models (Kuwajima et al. 2019). For example, a developer can visualize an NN's focus points for an incorrect inference and understand what was wrong, before additional training data is collected according to the analysis. If a machine learning model outputs an incorrect inference, but the visualized focus area is natural for humans, then an inaccurate ground truth label is suggested.

3.3.1 Related works and research directions for verification of machine learning models

In the field of theoretical computer science, the automatic design verification (Lam 2008) based on formal verification technologies for certain properties, such as safety and liveness (Garcia 2006), makes the verification of a machine learning model possible. Several automatic verification techniques exist for NNs and we categorize them here. The initial categories are function and decision problems. The former quantifies the degrees of properties, while the latter identifies if the properties are satisfied in a machine learning model. Related works for function problems address adversarial frequency and severity (Bastani et al. 2016) as well as maximum perturbation bound (Cheng et al. 2017), referring to the frequency of AE found, the expectation of the closest AE, and the maximum absolute value of the perturbation of inputs that do not change the outputs, respectively. Decision problems are further subdivided into verification and falsification, which seek a complete proof and counterexamples by best effort, respectively. Related works of verification are global safety (Pulina 2012), local safety (Pulina 2010), (ϵ, δ)-robustness (Katz et al. 2017), and (x, η, δ)-safe (Huang et al. 2017). Global safety is output bound, and local safety is the consistency of inference among close data points. A related example for falsification is the CNN Analyzer (Dreossi et al. 2017, 2019). It identifies counterexamples against the signal temporal logic (Donzé 2013) properties of in-vehicle automated driving systems and counterexamples of object (vehicle) detection by convolutional NNs. Further, Reluplex (Katz et al. 2017) is a solver used to both verify and falsify first-order propositional logics (Andrews 2002) against NNs using Rectified Linear Units (ReLU) (Nair 2010) for activation functions. Reluplex is an SMT (satisfiability modulo theories) solver (De Moura 2008) to verify properties of deep NNs or provide counterexamples against them by utilizing the simplex method (Dantzig 1987) and the partial linearity of the ReLU function. Dependability metrics set for NNs is a related work that proposes metrics such as scenario coverage, neuron activation pattern, interpretation precision for RICC (robustness, interpretability, completeness, and correctness) criteria (Cheng and Huang et al. 2018).

4 Engineering machine learning systems

In this section, we review open engineering problems in terms of the system level of in-vehicle automated driving systems as an example of safety-critical machine learning systems. Problems related to machine learning systems originate from machine learning models and the open environments in which automated vehicles function. The former is low modularity of machine learning systems due to the characteristics of machine learning models, such as lack of design specifications and lack of robustness. The latter include capturing physical operational environments and user behaviors of in-vehicle automated driving systems for requirements and addressing the intractableness of field operation testing (FOT) for verification. An open environment problem is not directly related to machine learning, although it is an important challenge for in-vehicle automated driving systems. In this paper, we consider open environments to be a common challenge for machine learning systems because machine learning models are employed to capture these complex environments.

4.1 Requirements of machine learning systems

In order to develop high quality systems and products, comprehensive requirements specifications and the evaluation of machine learning systems based on the requirements specification are needed; in turn, these require appropriate quality characteristics for the systems that can be used for requirements and evaluations. Quality characteristics of machine learning "systems" are more important in the context of industry than those of machine learning "models," because machine learning models are not used in a stand-alone manner but are always embedded in systems. System and software quality models have been developed for years; however, to the best of our knowledge, there is no standard quality model that adapts the characteristics of machine learning models—such as lack of requirements specifications, design specifications, interpretability, and robustness—into account. Thus, we conduct a gap analysis on a conventional system and software quality standard, SQuaRE (ISO 2014) in Sect. 5.

Another important aspect of machine learning (or any other) systems is that they cannot operate in every environment and require limitations or warranty scopes. Thus, a particular machine learning system must be implemented for a predefined environment. Environment attributes to be predefined for automated driving systems are static conditions such as weather, times of day, scene, road, as well as dynamic conditions (dynamics of) such as the vehicle under control and other moving objects (surrounding vehicles and pedestrians). However, there are various (uncountable) types of roads, traffic lights, and traffic participants, such as other vehicles (be they automated or manually driven) and pedestrians; therefore, it is not easy to define the operational environment for in-vehicle automated driving systems. An open engineering problem in the requirements specification of machine learning systems is that there is no standard means to design and define such environments, i.e., requirements specification cannot be clearly defined. In the automotive industry, this is called the operational design domain (Administration NHTS of Transportation UD 2017; Government U 2018) and it can be defined by conditions such as geographical areas, road types, traffic conditions, and maximum speed of the subject vehicle (Colwell et al. 2018).

4.1.1 Related works and research directions for requirements of machine learning systems

The German PEGASUS project is a joint initiative of vehicle manufacturers, suppliers, tool vendors, certification organizations, and research institutes, aiming to define standard quality assurance methods for automated-driving systems (Lemmer 2017). The purpose of this project is to clarify the expected performance level and evaluation criteria of automated driving systems through scenario-based verification. The scope of the project includes standard test procedures, continuous and flexible tool chains, the integration of tests into development processes, cross-company test methods, requirement definition methods, driving scenarios, and a common database of scenarios. Scenarios are collected from test drives and the market to demonstrate that systems are equal to, or better than, human drivers. Scenario collection, i.e., building requirements specification, and scenario-based verification are conducted in a continuous manner. Regular scenarios are continuously tested by simulation, and critical scenarios are tested through artificially configured environments on test courses. The PEGASUS project is an excellent example of the continuous requirements and verification for in-vehicle automated driving systems and their verification.

4.2 Design of machine learning systems

An open engineering problem at the system level of machine learning systems is designing systems that include machine learning models by considering and applying the characteristics of "Change Anything Change Everything" (CACE) (Sculley et al. 2015). CACE originates from a lack of design specification in machine learning models. Machine learning models are trained in a data-driven manner, thereby making the localizing of change difficult. If a small part is changed, then the entire machine learning changes once it is trained again. Subsequently, machine learning systems have to be changed for the newly trained machine learning models. In order to prevent reworking after training machine learning models, it is necessary to have system architectures that can cope with additional requirements without modification of the model.

In general, it is difficult for a machine learning model to achieve 100% accuracy on test data (Domingos 2012) and as its accuracy approaches 100%, further performance improvement becomes difficult. Therefore, optimizing machine learning models is not the only means to improve subsystem performance, thereby making a rigorous breakdown of subsystem requirements into machine learning model requirements essential for safety-critical machine learning systems. In this process, safety analysis methods and processes are important, such as the encapsulation of machine learning models by rule-based safeguarding and the use of redundant and diverse architecture that absorbs and mitigates the uncertainty of machine learning models.

4.2.1 Related works and research directions for design of machine learning systems

To the best of our knowledge, we do not find special techniques that directly address the design of machine learning systems. SOTIF (Wendorff 2017), a safety standard/process concerning performance limits of functionalities, focuses on securing functionalities with uncertainty. Uncertain functionalities include machine learning models. SOTIF has a process that includes identification of scenarios that can trigger unsafe actions (triggering conditions) for the system and system modifications to address them (Czarnecki 2018). The process standards can be potentially effective in the design of machine learning system in general, rather than evaluating an entire machine learning system upon completion of development. In addition to process standards, research directions include test stubs for machine learning models, encapsulation of machine learning models by rule-based safeguarding, and the use of redundant and diverse architecture that mitigates and absorbs the low robustness of machine learning models.

4.3 Verification of machine learning systems

The simplest approach to verifying an in-vehicle automated driving system is by verification against actual data. Accumulating a large number of safe automated driving trips, with long distances to match human drivers, will effectively demonstrate that in-vehicle automated driving systems are as safe as human drivers. In order to verify the system within a realistic time-frame, there are two options: reduce the required verification scenarios or accelerate the verification. Therefore, high accuracy verification models must be able to exclude unreal scenarios. It is necessary to accelerate simulation experimentation, thereby reproducing corner-case scenarios on test courses with a short mileage (i.e., scenarios with an extremely low probability of occurrence and ones that are difficult to statistically obtain through FOT on an actual road).

4.3.1 Related works and research directions for verification of machine learning systems

Obtaining statistically significant results would require FOT on a humongous number of miles (Kalra and Paddock 2016). Kalra and Paddock (2016) is based on a simple hypothesis testing, and the resulting required miles may not reflect actual situations. Research directions include building detailed close-to-reality models for driving scenes and scenarios to reflect the real world conditions and reduce FOT miles.

5 Quality of machine learning systems

We reviewed the open engineering problems in machine learning systems, and recognized that machine learning models are characterized by their lack of requirements specifications, design specifications, interpretability, and robustness. In this section, we study quality models for machine learning systems by discussing the combination of these machine learning characteristics and a conventional system and software quality standard, SQuaRE (ISO 2014).

5.1 Quality models for conventional systems

We focus on SQuaRE, ISO/IEC 25000 series (ISO 2014), as the conventional system quality baseline. Systems and software quality are usually studied in software engineering. One of the earliest work is an international standard ISO/IEC 9126 Software engineering-Product quality (ISO 2001), first issued in 1991. ISO/IEC 9126 classified software quality into six characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability. ISO/IEC 9126 was replaced by its succeeding international standard ISO/IEC 25000 series, Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE) (ISO 2014). SQuaRE is a widely acknowledged system quality standard and includes quality measures (QMs) and quality measure elements (QMEs) as well as quality models, characteristics, and sub-characteristics. These components have a tree structure (one-to-many relationships), and the top-level quality models are Product quality, Data quality, Quality in use, and IT service quality, as illustrated in Fig. 4. Boxes with thick lines and thin lines in Fig. 4 represent quality models and quality characteristics, respectively. Quality sub-characteristics are not defined for Data quality.

Each quality characteristic of Data quality, or each quality sub-characteristic of Product quality and Quality in use, has multiple QMs that define how to quantify the quality. A QM X is defined in the form of a formula, such that $X = A/B$ and $X = 1 - A/B$, and the elements in the formula A and B are QMEs. An example set of a quality model,

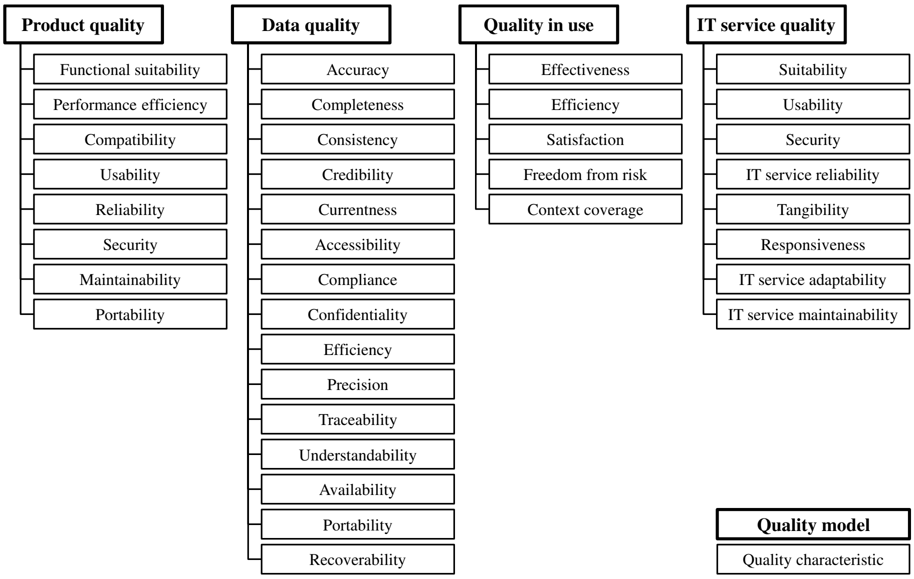


Fig. 4 Quality models and quality characteristics in SQuaRE

a characteristic, a sub-characteristic, a QM, and QMEs are Product quality, Reliability, Maturity, Mean time between failure, and Operation time (QME 1) and Number of system/software failures that actually occurred (QME 2), respectively. There are other QMs for the sub-characteristic Maturity (such as Failure rate, whose QMEs are Number of failures detected during observation time and Duration of observation). QMs and QMEs are not defined for IT service quality.

5.2 Gap analysis

We performed a gap analysis between conventional system quality models and future system quality models for machine learning systems, given a conventional system and software quality standard SQuaRE and the characteristics of the machine learning models introduced in this paper. In order to conduct the most fine and precise analysis, we checked each QME (such as the number of systems/software failures that actually occurred) against each machine learning characteristic (such as a lack of robustness) to see if the QME was affected by the machine learning characteristic. If a QME in machine learning systems became immeasurable, as is the case with conventional systems, then the parent quality (sub-)characteristic would have gaps. IT service quality model was ignored in this gap analysis because it has no QME defined in the ISO/IEC 25000 series. Table 1 presents an example of impact analysis of characteristics of machine learning models and QMEs defined for Functional suitability in Product quality. Req, Des, Rob, and Tra are abbreviations for lack of requirements specification, design specification, robustness, and transparency, respectively. Functional suitability in Product quality was selected only to serve as an example, and corresponding analysis was conducted for all QMEs of all quality models, except for IT service quality model.

Table 1 Example impact analysis (functional suitability characteristic in product quality model)

QM	QME	Req	Des	Rob	Tra
Functional coverage	Number of functions missing				
	Number of functions specified				
Functional correctness	Number of functions that are incorrect		†		
	Number of functions considered			‡	
Functional appropriateness of usage objective	Number of functions missing or incorrect among those that are required for achieving a specific usage objective		††		
	Number of functions required for achieving a specific usage objective			‡‡	
Functional appropriateness of system	Appropriateness score for a usage objective				
	Number of usage objectives				

† When the input changes slightly, the result can changes drastically. We cannot measure the correctness of the function precisely. Perturbed trials can quantify the uncertainty

‡ Functions considered cannot be defined strictly. For example, there are many pedestrian variations of pedestrian detection for an auto emergency braking (AEB) function, and it can be multiple functions. We cannot define functions without ambiguity

†† When the input changes slightly, the result can changes drastically. We cannot measure the correctness of the function precisely. Perturbed trials can quantify the uncertainty

‡‡ Functions considered cannot be defined strictly. For example, there are many pedestrian variations for pedestrian detection, and it can be multiple functions. We cannot define functions without ambiguity

We examined 1464 combinations of 366 QMEs and 4 characteristics of machine learning models to obtain the results. The number of combinations we identified as being affected by machine learning models was 20 from among 1464. Tables 2, 3, and 4 are the summaries of impact analysis on Product quality, Data quality, and Quality in use models affected by the characteristics of machine learning models. QM and QME levels are omitted. Each QME associated with a quality (sub-)characteristic was examined to determine if it was affected by any machine learning characteristics: a lack of requirements specification (Req), a lack of design specification (Des), a lack of robustness (Rob), or and lack of transparency (Tra). The section signs with numbers in parentheses next to machine learning characteristics are the indices to the itemization in the subsequent paragraphs. The number of QMEs affected by the machine learning characteristics are presented in the abovementioned tables. If we consider that the ratios of QMEs affected by characteristics of machine learning models are an indication of the impacts to quality (sub-)characteristics, then at the quality-model level, it is evident that the impact to Product quality is the highest, while those of Data quality and Quality in use are low.

The characteristics of machine learning models that affected QMEs the most were a lack of requirements specification and a lack of robustness. First, we discuss the impact of a lack of requirements specification. Quality characteristics involving preconditions (such as operational contexts, the interval of values, and operational environments) were affected by a lack of requirements specification. This is because requirements specifications define preconditions for systems. As discussed previously, machine learning models are trained using data-driven processes and lack explicit requirements specifications. Instead, preconditions are implicitly encoded in training data and not explicitly described. Thus, the following QMEs become unmeasurable due to a lack of preconditions (requirements specifications).

Table 2 Impact analysis on product quality model

Characteristic	Sub-characteristic	Number of QMEs		
		All	Affected	
Functional suitability	Functional correctness	2	2	Req (§5), Rob (§9)
	Functional appropriateness	4	2	Req (§6), Rob (§10)
	Others	2	0	
Performance efficiency	All	29	0	
Compatibility	All	8	0	
Usability	Operability	18	1	Tra (§19)
	Others	25	0	
Reliability	Maturity	8	2	Rob×2 (§12, §13)
	Fault tolerance	7	2	Rob (§11), Des (§16)
	Others	8	0	
Security	All	22	0	
Maintainability	Modularity	4	2	Tra (§17)
	Analysability	6	1	Tra (§20)
	Modifiability	7	1	Des (§18)
	Testability	6	1	Rob (§14)
	Others	4	0	
Portability	Adaptability	6	1	Req (§1)
	Replaceability	8	2	Req (§7)
	Others	5	0	
Total		179	15	

§1 Number of functions which were tested in different operational environments
[Product quality / Portability / Adaptability / Operational environment adaptability]
§2 Number of data items for which can be defined a required interval of values
[Data quality / Accuracy / Data accuracy range]
§3 Total number of required distinct contexts of use
[Quality in use / Context coverage / Context completeness / Context completeness]
§4 Total number of additional contexts in which the product might be used
[Quality in use / Context coverage / Flexibility / Flexible context of use]

Note that the corresponding quality model, characteristic, sub-characteristic, and QM are described in square brackets. Different operational environments in which systems must be tested, required intervals of values for data items, distinct contexts of use, and additional contexts in which the product might be used cannot be defined for data-driven training processes; the above QMEs are not measurable.

The reasons for a lack of requirements specifications in machine learning models are twofold: a lack of preconditions (introduced in the last paragraph) and a difficulty defining the desired behaviors of machine learning models due to the wide variety of input and output patterns. For example, there are numerous variations of pedestrians (such as young and old, one with bags and umbrella) for an AEB function and it is difficult to define the function precisely (the types of pedestrians that the system covers) without ambiguity. Being unable to define precise functions affects Function suitability, as well as Portability

Table 3 Impact analysis on data quality model

Characteristic	Number of QMEs		
	All	Affected	
Accuracy	14	3	Req ×2 (§2, §8), Rob (§15)
Completeness	16	0	
Consistency	12	0	
Credibility	8	0	
Currentness	6	0	
Accessibility	6	0	
Compliance	4	0	
Confidentiality	4	0	
Efficiency	14	0	
Precision	4	0	
Traceability	6	0	
Understandability	14	0	
Availability	6	0	
Portability	6	0	
Recoverability	6	0	
Total	126	3	

Table 4 Impact analysis on quality in use model

Characteristic	Sub-characteristic	Number of QMEs		
		All	Affected	
Effectiveness	All	8	0	
Efficiency	All	11	0	
Satisfaction	All	13	0	
Freedom from risk	All	21	0	
Context coverage	Context completeness	2	1	Req (§3)
	Flexibility	6	1	Req (§4)
Total		61	2	

of Product quality. Being unable to define precise normal conditions, outliers for a wide variety of input data values are not definable, neither. The following QMEs are not measurable due to the difficulty of defining behaviors:

§5 Number of functions that are incorrect
[Product quality / Functional suitability / Functional correctness / Functional correctness]

§6 Number of functions missing or incorrect among those that are required for achieving a specific usage objective
[Product quality / Functional suitability / Functional appropriateness / Functional appropriateness of usage objective]

§7 Number of functions which produce similar results as before
[Product quality / Portability / Replaceability / Functional inclusiveness]
§8 Number of data values that are outliers
[Data quality model / Accuracy / Risk of data set inaccuracy]

Next, we discuss the impact of a lack of robustness. QMEs that observe machine learning system behavior are affected by a lack of robustness. When the inputs of machine learning models change even slightly, the results can change drastically. Therefore, the behavior of such systems becomes uncertain and we cannot measure (count) correct behavior. Moreover, we noticed that the QMEs affected by low robustness were similar to those affected by a lack of requirements specification. The QMs using these QMEs are typically ratios, with numerators being QMEs that count correct behavior and denominators being QMEs that count preconditions. For example, one of the quality measures of Functional correctness is $X = 1 - A/B$, where $A =$ [Number of functions that are incorrect], $B =$ [Number of functions considered]. We cannot measure the numerator A and the denominator B due to the two characteristics of machine learning models—a lack of robustness and a lack of requirements specification, respectively. The following QMEs are not precisely measurable due to a lack of robustness:

§9 Number of functions that are incorrect
[Product quality / Functional suitability / Functional correctness / Functional correctness]
§10 Number of functions missing or incorrect among those that are required for achieving a specific usage objective
[Product quality / Functional suitability / Functional appropriateness / Functional appropriateness of usage objective]
§11 Number of avoided critical and serious failure occurrences based on test cases
[Product quality / Reliability / Fault tolerance / Failure avoidance]

QMEs related to negative events affected the difficulty of capturing rare cases of machine learning models, which is another form of a lack of robustness. Outliers and failures in SQuaRE should have included rare cases; however, rare cases may not appear in a limited time frame and when they do, the extremely low probability of occurrence may be neglected. As mentioned previously, an extremely long FOT is required to capture such rare events. The following are the QMEs that were underestimated due to the difficulty of overlooking rare cases:

§12 Number of system/software failures actually occurred
[Product quality / Reliability / Maturity / Mean time between failure, MTBF]
§13 Number of failures detected during observation time
[Product quality / Reliability / Maturity / Failure rate]
§14 Number of test functions required
[Product quality / Maintainability / Testability / Test function completeness]
§15 Number of data values that are outliers
[Data quality / Accuracy / Risk of data set inaccuracy]

There is a small impact on machine learning systems due to the lack of design specification and lack of transparency characteristics. If there are no design specifications, we cannot estimate the effort of a system modification nor the impact of a local modification to the overall system. We cannot forecast how many hours the training process will require, in advance. In addition, we cannot know the strengths and weaknesses of automatically trained machine learning models in general. Therefore, we cannot know the redundancy of components without design specification or transparency. Models with similar weaknesses do not work as redundancies, and redundant installation does not make sense for machine learning models. The following are QMEs that are unmeasurable due to a lack of design specifications and a lack of transparency:

§16 Number of system components redundantly installed
[Product quality / Reliability / Fault tolerance / Redundancy of components]
§17 Number of components which are implemented with no impact on others
[Product quality / Maintainability / Modularity / Coupling of components]
§18 Expected time for making a specific type of modification
[Product quality / Maintainability / Modifiability / Modification efficiency]

Since there is no established method of diagnostic and monitoring functionalities for machine learning models, the following QMEs are not measurable for machine learning systems.

§19 Number of functions having state monitoring capability
[Product quality / Usability / Operability / Monitoring capability]
§20 Number of diagnostic functions useful for causal analysis
[Product quality / Maintainability / Analysability / Diagnosis function effectiveness]

We have discussed the combination of machine learning characteristics with a conventional system and software quality standard, SQuaRE. The typical gaps for the quality models of machine learning systems were found in requirements specification (precondition specification and level of detail for function specification) and robustness (uncertainty of observation and extremely low probable rare cases). In order to address these gaps, system quality models can be modified and/or extended. We introduce the direction to address these gaps in Sect. 5.3.

5.3 Toward quality models for machine learning systems

The first set of challenges exist in quality measures for preconditions and functions (functionalities) for machine learning systems, that is, requirements specification. We assume that preconditions and function specifications are defined by input range and pairs of input/output, respectively. If input and/or output data are high-dimensional, both defining preconditions and detailed function specifications are difficult. As machine learning models are trained in a data-driven manner, we inevitably conclude that data is involved. One natural idea is first to manually engineer the deductive specifications in as detailed a manner as possible and second to prepare data that includes example instances for requirements specifications. Requirements specifications of machine learning systems cannot fully define the preconditions and functions; however, the remaining uncertainty of specifications is covered by examples. In order to make requirements specifications as detailed as possible, we need quality definitions (and subsequently QM) of requirements specifications themselves.

A type of QM for requirements specifications is the sum of the quality of deductive requirements specification and the quality of inductive requirements specification, that is, sample data.

The quality of deductive requirements specifications for machine learning systems—that is, the level of details of requirements specifications—is not straightforward to measure. Although not quantitative, a proxy of quality of deductive requirements specification is to measure the level of detail of the background argument. An earlier study (Ishikawa and Matsuno 2018) used structured arguments, like goal structure notation (GSN) (Kelly 2004), to address uncertain requirements and environments. Quality measures of deductive requirements specification such as the following can be added to quality models of machine learning systems:

– "Number of functions with preconditions specified with structured argument" divided by "Number of functions that could benefit from specifying preconditions"
– "Number of functions with detailed function specification with structured argument" divided by "Number of functions that could benefit from detailed function specification"

Further, the quality of inductive requirements specification (sample data) must be defined as the coverage of deductive requirements specifications, that is, how much deductive requirements specifications were covered by the inductive requirements specification (sample data). If the structured argument is in a tree structure, the ratio of leaf nodes that have corresponding sample data can be a quality measure of sample data. A quality measure of inductive requirements specification is given by the following:

– "Number of GSN solutions (leaf nodes) having corresponding sample data" divided by "Number of GSN solutions (leaf nodes) that could benefit from specifying sample data"

It is also important to handle the uncertainty of observation of machine learning systems in the quality models for machine learning systems. The current quality measures are deterministic. Introducing a number of trials and variance to quality measures will incorporate the uncertainty of observation and improve the expression power of quality models.

Another aspect in the lack of robustness is the extremely low probability of rare cases. It must be noted that it is not possible to identify all rare cases by definition. We cannot evaluate the result of rare case discovery; however, we can see the quality of the process or the effort involved in it. Quality measures of the process of discovering rare cases would be the effort in rare case discovery plus the number of rare cases discovered in a unit of time.

A viewpoint that is not included in the current quality models is development data, although SQuaRE has Data quality. Data quality in SQuaRE is about the data included in the system itself, such as customer mailing address database. For machine learning systems, development data—that is, test and training data—is rather important, and the quality models for machine learning systems must include the corresponding quality. There are two different definitions for test data quality and training data quality. The former is related to inductive requirements specification, that is, an aforementioned quality of sample data. Test data quality includes the gap between manually engineered deductive requirements specifications and actually collected sample data points. The latter is related to design specification and includes the quality of manually annotated supervisory signals. This will be a trade-off to the cost of labor-intensive annotation processes.

6 Conclusion

With the rapid development of technology in recent years, machine learning has begun to be employed in various systems. To use machine learning in a safety-critical system, such as an automated driving system, it is necessary to demonstrate the safety and security of the system to society through the engineering process. In this paper, taking automated driving as an example, we presented open engineering problems with corresponding related works and research directions from the viewpoints of requirements, designs, and verifications for machine learning models and systems.

At the level of the machine learning model, we hypothesized an ideal training process that connects deductive requirements and data-driven training, thereby considering test data as a requirements specification and training data as a design specification. Moreover, we recognized that the characteristics of machine learning models are a lack of requirements specification, a lack of design specification, a lack of interpretability, and a lack of robustness. We also discussed the combination of a conventional system and software quality standard, SQuaRE, and the aforementioned characteristics of machine learning models to study the quality models for machine learning systems. It turned out that a lack of requirements specification (precondition specification and level of detail for function specification) and a lack of robustness (uncertainty of observation and extremely low probability rare cases) have the largest impact on the conventional system quality models. Further, we discussed the direction of future quality models for machine learning systems; however, most of it is a subject for future research.

Future research directions include the development of element technologies for engineering machine learning models and systems, such as requirements specification techniques to cover test data distribution or open environments. As evident from this paper, there are numerous open engineering problems and possible directions to address them. However, to establish an engineering process for safety-critical machine learning systems, even if each company individually performs its own engineering processes based on its own concepts, process activities and work products cannot be automatically accepted by human society. Individual practices are not standard, and in order to achieve accountability, need evaluation on a case-by-case basis by a third party, particularly in case of problems. In such evaluations, own engineering practices of individual companies are at risk for being misunderstood, otherwise proprietary development information has to be disclosed for accountability. Thus, we need widely accepted standards to avoid these situations. Attempts to research element technologies along with standard guidelines for requirements, designs, and verifications would also be practically helpful. For example, a standard guideline for multiple verification tiers (actual data testing for normal conditions, simulated data testing for the corner cases, automatic verification for highest integrity levels only, falsification in middle integrity levels, etc.) would encourage the practical use of verification techniques and help an industry suffering from a lack of quality assurance of machine learning systems. Another approach is to develop standard quality models for machine learning systems. In this paper, we discussed the quality models for machine learning systems based on SQuaRE. Future research directions include discussing quality characteristics beyond SQuaRE, defining specific QM and QME, and quality characteristics and sub-characteristics if necessary.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Administration NHTS of Transportation UD, (2017). Automated driving systems: A vision for safety 2.0.

Ali, G. G. M. N., & Chan, E. (2011). Co-operative data access in multiple road side units (rsus)-based vehicular ad hoc networks (VANETS). In *Proceedings of the Australasian telecommunication networks and applications conference, ATNAC 2011, Melbourne, Australia, November 9–11, 2011* (pp. 1–6). IEEE. https://doi.org/10.1109/ATNAC.2011.6096651.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. CoRR arXiv:1606.06565.

Andrews, P. B. (2002). *Introduction to mathematical logic and type theory: To truth through proof* (2nd ed.). Norwell, MA: Kluwer Academic Publishers.

Arlot, S., Celisse, A., et al. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, *4*, 40–79.

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., & Criminisi, A. (2016). Measuring neural net robustness with constraints. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 2613–2621). Red Hook: Curran Associates Inc.

Ben-David, S., Kushilevitz, E., & Mansour, Y. (1997). Online learning versus offline learning. *Machine Learning*, *29*(1), 45–63. https://doi.org/10.1023/A:1007465907571.

Bickel, S., Brückner, M., & Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research.*, *10*, 2137–2155. https://dl.acm.org/citation.cfm?id=1755858.

Binder, A., Montavon, G., Lapuschkin, S., Müller, K., & Samek, W. (2016). Layer-wise relevance propagation for neural networks with local renormalization layers. In A. E. P Villa, P. Masulli, A. J. P. Rivero (Eds.), *Proceedings artificial neural networks and machine learning-ICANN 2016- 25th international conference on artificial neural networks*, Barcelona, Spain, September 6–9, 2016, Part II, Lecture Notes in Computer Science. (vol. 9887, pp. 63–71). Springer. https://doi.org/10.1007/978-3-319-44781-0_8.

Bird, S., Crankshaw, D., Gibson, G., Gonzalez, J., Lakshmiratan, A., Li L.E., Re, C., & Sen, S. (2017). In *Proceedings of the workshop on ml systems at nips 2017*.

Borraz, R., Navarro, P. J., Fernández, C., & Alcover, P. M. (2018). Cloud incubator car: A reliable platform for autonomous driving. *Applied Sciences*. https://doi.org/10.3390/app8020303.

Cheng, C., Huang, C., & Yasuoka, H. (2018) Quantitative projection coverage for testing ML-enabled autonomous systems. In S. K. Lahiri, C. Wang (Eds.) *Proceedings of automated technology for verification and analysis - 16th international symposium, ATVA 2018*, Los Angeles, CA, USA, October 7-10, 2018, Lecture Notes in Computer Science . (vol. 11138, pp. 126–142). Springer. https://doi.org/10.1007/978-3-030-01090-4_8.

Cheng, C., Nührenberg, G., Huang, C., Ruess, H., & Yasuoka, H. (2018). Towards dependability metrics for neural networks. In *Proceedings of 16th ACM/IEEE international conference on formal methods and models for system design, MEMOCODE 2018*, Beijing, China, October 15–18, 2018, (pp. 43–46). IEEE, https://doi.org/10.1109/MEMCOD.2018.8556962.

Cheng, C., Nührenberg, G., & Yasuoka, H. (2018). Runtime monitoring neuron activation patterns. CoRR abs/1809.06573, arXiv:1809.06573.

Cheng, C. H., Nührenberg, G., & Ruess, H. (2017). Maximum resilience of artificial neural networks. In *ATVA*. Springer, Cham.

Colwell, I., Phan, B., Saleem, S., Salay, R., & Czarnecki, K. (2018). An automated vehicle safety concept based on runtime restriction of the operational design domain (pp. 1910–1917). https://doi.org/10.1109/IVS.2018.8500530.

Czarnecki, K. (2018). On-road safety of automated driving system (ads)—Taxonomy and safety analysis methods. https://doi.org/10.13140/RG.2.2.28313.93287.

Dantzig, G. B. (1987). Origins of the simplex method. Tech. rep.: stanford univ ca systems optimization lab.

De Moura, L., & Bjørner, N. (2008). Z3: An efficient smt solver. In *Proceedings of the theory and practice of software, 14th international conference on tools and algorithms for the construction and analysis of systems*, (pp. 337–340). Springer, Berlin. TACAS'08/ETAPS'08, http://dl.acm.org/citation.cfm?id=1792734.1792766.

Domingos, P. (2012). A few useful things to know about machine learning. *Commun ACM*, *55*(10), 78–87. https://doi.org/10.1145/2347736.2347755.

Donzé A (2013) On signal temporal logic. In *Proceedings of the international conference on runtime verification*, (pp. 382–383). Springer, Berlin.

Dreossi, T., Ghosh, S., Sangiovanni-Vincentelli, A. L., & Seshia, S.A., (2017). Systematic testing of convolutional neural networks for autonomous driving. In *Proceedings of the ICML workshop on reliable machine learning in the wild*.

Dreossi, T., Donzé, A., & Seshia, A. S. (2019). Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning*. https://doi.org/10.1007/s10817-018-09509-5.

Elkahky, A. M., Song, Y., & He, X. (2015). A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th international conference on world wide web, international world wide web conferences steering committee, Republic and Canton of Geneva, Switzerland, WWW '15*, (pp. 278–288). https://doi.org/10.1145/2736277.2741667.

Falcini, F., & Lami, G. (2017). Deep learning in automotive: Challenges and opportunities. In A. Mas, A. Mesquida, R. V. O'Connor, T. Rout, & A. Dorling (Eds.), *Software process improvement and capability determination* (pp. 279–288). Cham: Springer.

Garcia, F. A., & Sánchez, A. (2006) Formal verification of safety and liveness properties for logic controllers. a tool comparison.In *Proceedings of the 3rd international conference on electrical and electronics engineering*. (pp. 1–3).

Government U, of Transportation UD (2018) 2018 federal guide to self-driving cars and automated driving: preparing for the future of transportation—Automated vehicles 3.0 safety issues and role of the government in autonomous regulation.

Graves, A., Jaitly, N., & Mohamed, A. (2013). Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of the IEEE workshop on automatic speech recognition and understanding*, Olomouc, Czech Republic, December 8–12, 2013, (pp. 273–278). IEEE https://doi.org/10.1109/ASRU.2013.6707742.

Grün, F., Rupprecht, C., Navab, N., & Federico, T. (2016). A taxonomy and library for visualizing learned features in convolutional neural networks. In *Proceeding of the ICML workshop on visualization for deep learning*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (pp. 770–778).

Huang, X., Kwiatkowska, M. Z., Wang, S., & Wu, M. (2017). Safety verification of deep neural networks. In *Proceedings of the CAV*.

INCOSE (2015). Systems engineering handbook: A guide for system life cycle processes and activities, version 4.0 edn. Hoboken: Wiley.

Ishikawa, F., & Matsuno, Y. (2018). Continuous argument engineering: Tackling uncertainty in machine learning based systems. In B. Gallina, A. Skavhaug, E. Schoitsch, & F. Bitsch (Eds.), *Computer safety, reliability, and security–SAFECOMP 2018 workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås,* Sweden, September 18, 2018, (vol. 11094, pp. 14–21). Proceedings, Springer, Lecture Notes in Computer Science. https://doi.org/10.1007/978-3-319-99229-7_2.

ISO 26262–1:2018, (2018). Road vehicles–functional safety-part 1: Vocabulary. International organization for standardization: Tech. rep.

ISO IEC 25000:2014. (2014). Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. International organization for standardization, international electrotechnical commission: Standard.

ISO IEC 9126:2001. (2001). Software engineering—product quality. International organization for standardization, international electrotechnical commission: Tech. rep.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An introduction to statistical learning: With applications in R*. Berlin: Springer.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM international conference on multimedia*, (pp. 675–678). ACM, New York, NY, USA, MM '14, https://doi.org/10.1145/2647868.2654889.

Kalra, N., & Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, *94*, 182–193. https://doi.org/10.1016/j.tra.2016.09.010.

Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Towards proving the adversarial robustness of deep neural networks. In L. Bulwahn, M. Kamali, & S. Linker (Eds.), *Proceedings of the first workshop on formal verification of autonomous vehicles (FVAV '17)*, electronic proceedings in theoretical computer science, (vol. 257, pp. 19–26). Turin, Italy. http://eptcs.web.cse.unsw.edu.au/paper.cgi?FVAV2017.3.

Katz, G., Barrett, C. W., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). In CAV. *Reluplex: An efficient smt solver for verifying deep neural networks*. Cham: Springer.

Kelly, T., & Weaver, R. (2004). The goal structuring notation—a safety argument notation. In *Proceedings of dependable systems and networks 2004 workshop on assurance cases*.

Koopman, P., & Wagner, M. (2016). Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, *4*, 15–24. https://doi.org/10.4271/2016-01-0128.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Red Hook: Curran Associates, Inc.

Kuwajima, H., Tanaka, M., & Okutomi, M. (2019). Improving transparency of deep neural inference process. *Progress in Artificial Intelligence*, *8*(2), 273–285. https://doi.org/10.1007/s13748-019-00179-x.

Lam, W. K. (2008). *Hardware design verification: simulation and formal method-based approaches* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Lemmer, K., & Mazzega, J. (2017). Pegasus: Effectively ensuring automated driving. In *VDA technical congress*.

Li, L. E., Dragan, A., Niebles, J. C., & Savarese, S. (2017). nips workshop on machine learning for intelligent transportation systems.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. CoRR abs/1312.5602

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Müller, K. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, *65*, 211–222. https://doi.org/10.1016/j.patcog.2016.11.008.

Murphy, K. P. (2013). *Machine learning: A probabilistic perspective*. Cambridge, MA: MIT Press.

Nair, V., & Hinton, G.E. (2010). Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz, & T. Joachims (Eds.) *Proceedings of the 27th international conference on machine learning (ICML-10)*, June 21–24, 2010, (pp. 807–814). Haifa: Omnipress. http://www.icml2010.org/papers/432.pdf.

Ng, A. (2015). Deep learning. In *nVIDIA GPU technology conference (GTC)*.

Poggenhans, F., Pauls, J., Janosovits, J., Orf, S., Naumann, M., Kuhnt, F., & Mayr, M. (2018). Lanelet2: A high-definition map framework for the future of automated driving. In *Proceedings of the ITSC*, (pp. 1672–1679). IEEE.

Pulina, L., & Tacchella, A. (2010). An abstraction-refinement approach to verification of artificial neural networks. In *Proceeding of the CAV*.

Pulina, L., & Tacchella, A. (2012). Challenging smt solvers to verify neural networks. *AI Communications*, *25*(2), 117–135.

Report of traffic collision involving an autonomous vehicle (ol 316). (2018). https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/autonomousveh_ol316+.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 1135–1144). San Francisco, CA, USA, August 13–17, 2016.

Salay, R., Queiroz, R., & Czarnecki, K. (2017). An analysis of ISO 26262: Using machine learning safely in automotive software. CoRR abs/1709.02435, arXiv:1709.02435.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Proceedings of the 28th international conference on neural information processing systems*, (Vol. 2, pp. 2503–2511). Cambridge, MA: MIT Press. NIPS'15.

Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences. CoRR abs/1605.01713

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the international conference on learning representations*.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, *36*(2), 111–133.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 3104–3112). Red Hook: Curran Associates, Inc.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., & Fergus, R. (2013). Intriguing Properties of Neural Networks. CoRR abs/1312.6199.

Tsymbal, A. (2004). The problem of concept drift: Definitions and related work. Tech. rep.

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., et al. (2019). Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, *18*(6), 463–477. https://doi.org/10.1038/s41573-019-0024-5.

VDA QMC Working Group 13/Automotive SIG (2015). Automotive spice process assessment/reference model version 3.0. Tech. rep. Automotive SPICE.

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*(4), 964–994. https://doi.org/10.1007/s10618-015-0448-4.

Wendorff, W. (2017). Quantitative sotif analysis for highly automated driving systems. In *Safetronic*.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., & Darrell, T. (2018). BDD100K: A diverse driving video database with scalable annotation tooling. CoRR abs/1805.04687, arXiv:1805.04687.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision-ECCV 2014* (pp. 818–833). Cham: Springer.

Zendel, O., Murschitz, M., Humenberger, M., & Herzner, W. (2015). CV-HAZOP: Introducing test data validation for computer vision. In *Proceedings of the international conference on computer vision, ICCV 2015*, Santiago, Chile, December 7–13, 2015, IEEE Computer Society. (pp. 2066–2074). https://doi.org/10.1109/ICCV.2015.239.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. CoRR abs/1611.03530, arXiv:1611.03530.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2015). Object detectors emerge in deep scene cnns. In *Proceedings of the international conference on learning representations*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.