Open Questions in Testing of Learned Computer Vision Functions for Automated Driving

Matthias Woehrle, Christoph Gladisch(✉), and Christian Heinzemann

Robert Bosch GmbH, Corporate Research,
Robert-Bosch-Campus 1, 71272 Renningen, Germany
{Matthias.Woehrle,Christoph.Gladisch,Christian.Heinzemann}@de.bosch.com

Abstract. Vision is an important sensing modality in automated driving. Deep learning-based approaches have gained popularity for different computer vision (CV) tasks such as semantic segmentation and object detection. However, the black-box nature of deep neural nets (DNN) is a challenge for practical software verification. With this paper, we want to initiate a discussion in the academic community about research questions w.r.t. software testing of DNNs for safety-critical CV tasks. To this end, we provide an overview of related work from various domains, including software testing, machine learning and computer vision and derive a set of open research questions to start discussion between the fields.

1 Introduction

Deep learning-based approaches have achieved impressive performance results on a wide range of benchmarks in various domains such as computer vision [40]. As industrial application of deep neural networks (DNNs) increases, there is an increased need for verification and validation (V&V). This paper focuses on practical verification and concretely on testing of computer vision (CV) software. The Software under Test is a CV task, such as object detection and semantic segmentation, embedded in an automotive camera.[1] For simplicity, we focus on stateless CV functions that evaluate each image individually. Our concrete application domain is autonomous driving (AD), so the software may be safety-relevant.

We focus on *falsification of the software* during development, *i.e.* identifying defects in the software early in the development cycle, rather than validation-specific topics such as distributional shifts or global performance characteristics. Our verification context is testing of DNNs that extract information from individual images, *e.g.* objects and their bounding boxes or annotate pixel-wise semantic labels, as shown in Fig. 1. This is challenging as the input space for

[1] Many verification techniques are up to now only applied to image classification. While this simpler CV task is not relevant for our application, the corresponding methods are good starting points for further study.

© Springer Nature Switzerland AG 2019
A. Romanovsky et al. (Eds.): SAFECOMP 2019 Workshops, LNCS 11699, pp. 333–345, 2019.
https://doi.org/10.1007/978-3-030-26250-1_27

a typical CV function is vast, *e.g.* for a Cityscapes [8] image approximately $10^{5681751}$ inputs would be theoretically possible. While the subspace of relevant images is many orders of magnitudes lower than this astronomical number, there is no natural constraint or generative model available from which we can sample tests incl. ground truth as we could from a test input model such as a formal grammar [3].[2] For test specification, we additionally need to consider checking of the CV function output. Getting ground truth labels via manual labeling is a labor- and time-consuming and expensive task [10, 19], *e.g.* for the Cityscapes semantic segmentation task it required on average more than 1.5 hours for the 5000 images [8].

The open and real-world context of AD and the high-dimen-sional and unstructured input space of computer vision exacerbate the oracle problem of software testing [3] leaving us with the question: *How do we create good (relevant and meaningful) test data efficiently for a CV function interpreting images of driving scenes in the physical world? How do we verify relevant properties of the corresponding DNNs?*

We are interested in the question how one can ensure that a DNN works correctly based on a finite test set (*cf.* [40])? Previous work has surveyed the general topics of testing of machine learning (ML) software considering both implementation and conceptual issues [5]. Additionally, Borg *et al.* [4] review V&V challenges for ML in the automotive industry. However, due to our specific application domain and CV task, our verification context differs considerably. The contribution of this paper is a discussion of approaches that may contribute to obtaining a good test set for DNNs and CV functions in the automated driving context.

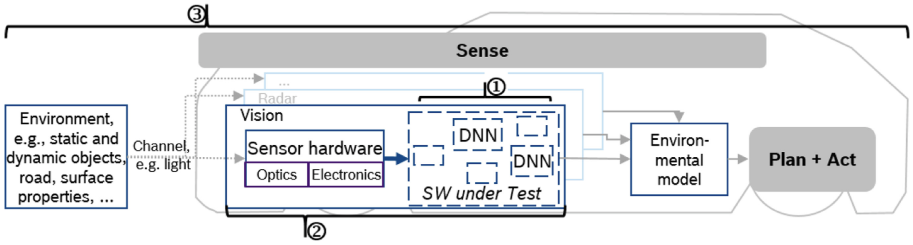

Fig. 1. Overview of the V&V task detailing on the Software (SW) under test and its system embedding with 3 exemplary test setups.

Note that in an industrial development process, testing is performed throughout various development stages in complementary ways to provide an overall argument that the system implementation is correct. To this end, various (*i*) test methods and (*ii*) test setups are used. First, different test methods are used to address various test concerns such as checking for implementation errors, security aspects, data pre-processing issues, labeling errors, leakage, quantization

[2] We discuss first steps in this direction in the context of synthetic data in Sec. 2.3.

issues, timing and consistency constraints, robustness guarantees and satisfaction of requirement specifications. Second, as examples for different test setups, Fig. 1 depicts three examples using curly braces. The smallest brace (1) concerns isolated testing of CV functions that we focus on in the following, where test images are directly processed by the SW under Test (a DNN). As we can see in the figure, we can also enlarge the test setup, *e.g.* (2) include the sensor hardware in our test setup (either in a real hardware-in-the-loop setup or using a model thereof) or (3) perform end-to-end testing [33]. Note that for (3), we perform closed-loop testing as the loop is closed through the vehicle interacting with the real world.

In the following, we discuss selected approaches from software testing, machine learning and computer vision and present several open exemplary research questions to initiate a discussion. Due to space limitations, further topics such as model explainability and interpretability, debugging of ML models, software, system and ML safety, AD V&V, modeling and simulation, (automatic) labeling, data science, data cleaning and exploratory data analysis are omitted. Section 2 presents three approaches for generating test inputs. First, we consider sampling around test images (Sect. 2.1). The most prominent approach in this category is generating (minimal) adversarial examples. Second, we consider analysis approaches that characterize relevant and important factors to be considered for a good test set (Sect. 2.2). Third, we consider testing based on synthetic test data (Sect. 2.3).

For test evaluation (Sect. 3), we discuss ground truth as well as relevant properties and metrics. Here, we need to differentiate the general approach of testing guided by a particular test concern such as checking for average case behavior (*e.g.* IoU on a test set [8]), exceptional behavior (*e.g.* uncertainty assessment in edge cases), and specifications in particular use cases (*e.g.* detection of relevant, far-away objects). Note that recent academic work has focused on comparing average case behavior based on cost metrics - mainly to evaluating competing designs - while verification and testing is typically concerned with (worst-case) behavior w.r.t. specific properties.

2 Test Input Generation

2.1 Sampling Around Labeled Test Images

One input model used for testing DNNs is to sample around labeled example images. Sampling is performed by considering some distance metric around a reference image, *i.e.* only on the input and independent of the CV task.

Adversarial Data Generation. Adversarial examples for image classification is a main research topic, where the input model is sampled around a given image under the assumption that the label stays the same as long as the distance of the new sample is below a selected threshold. A typical norm in the context of (minimal) adversarial examples are L_p norms, *e.g.* L_∞ used in [35,38]. Such a metric integrates well into current deep learning frameworks and thus can be used to generate tests efficiently. However, for autonomous driving and other notions of robustness, other distance metrics may be more relevant, *e.g.* based on noise characteristics of the imager. An impressive in-depth overview of algorithms for verifying neural networks for image classification w.r.t. adversarial robustness is [23]. The authors detail on several algorithms, classify them into an intuitive framework, and detail on soundness and completeness of the approaches. However, these techniques cannot be directly applied to CV tasks for automated driving, *e.g.* due to (i) scalability constraints and (ii) their focus on image classification. In a generalization for verification of non-linear properties, it is noted that defining a relevant input set is a hard task and a concrete first step is to consider sampling sets around test examples (using *e.g.* an L_p norm) and perform "weaker verification" [29]. Apart from formal worst-case analyses of minimal, additive adversarial perturbations, there are various notions of robustness [10], *e.g.* datasets considering robustness to corruption and common perturbations [16] as well as computer vision hazards [41].

Concolic Testing and "Exhaustiveness". Concolic test generation, *e.g.* [32], is a white-box software test generation technique which extends symbolic execution of source code using concrete test executions to remedy that source code may not be available or too complex for symbolic execution. Concolic testing allows generating an exhaustive test suite based on a coverage criterion, such as branch coverage. There is first work to apply concolic testing to deep learning functions [36], which is based on input coverage on the adversarial input model described above. The major concern for any method trying to completely sample the input space is that only a subspace of the vast input space is actually functionally relevant and this space is very difficult to characterize. Cheng *et al.* [7] argue for learning (parts of) an input space representation alongside the function. However, this would be a discriminative model not a generative one.[3] Apart from generating test inputs, we also need to provide corresponding oracles for new test inputs. Apart from top-1 label invariance within a threshold described above it is unclear what such an oracle would look like for concolic testing.

Data Augmentation. Data augmentation for CV functions leverages a large variety of transformations on images like rotations and flips. While its main purpose is multiplication of training data, these transformations can also be leveraged as a basis for testing. Data augmentation may either be integrated into machine learning frameworks or used with dedicated libraries [18]. In these libraries, transformation of inputs is coupled with corresponding transformation of ground truth, e.g. of a segmentation map. Some data augmentation techniques may be even amenable to a formal analysis such as the support of rotations in DeepPoly [35]. Augmentation techniques are obviously only an approximation, since transformations in the real world are much more complex than the transformation in the image space, *e.g.* considering illumination. Additionally, realistic parameterization of image augmentations including thresholds and equivalence

[3] With the discriminative model we could only dismiss generated irrelevant test inputs, while with a generative model, we could directly generate relevant tests.

classes is a major concern for verification. As an example, we might not use horizontal flipping as this is outside the Operational Design Domain (*cf.* Sect. 2.2) of the function, but may use vertical flipping to represent left- and right hand traffic. Based on this, we can identify a difference in augmentation for training versus verification. In training, on-line augmentation with random transformations is used to sample the input space sparsely for a large training set. Verification may rather focus on a small set of important test images where augmentation is performed densely, *e.g.* to better characterize robustness.

Summary. All the discussed methods sample around individual test images and indiscriminately modify images independently of concrete image content. However, image-specific modifications can also be considered for testing: Shetty *et al.* [34] consider image specific object removals and Yuille *et al.* check the effect of occluders on object detection [40]. Some of the discussed methods aspire exhaustive sampling and formal verification, however all results are local around individual test samples.

Exemplary Research Questions. (*i*) What notions of robustness and corresponding test images should be included in a good test set? (*ii*) Which kind of coverage criterion could be used to argue exhaustiveness of a test set? (*iii*) What data augmentations should be used on images in a good test set?

2.2 Domain and Data Analysis

Domain and data analysis creates and leverages additional information of the relevant context. This includes an analysis of relevant inputs, *e.g.* analyze distributions in pixel space, and an analysis of the outputs, *e.g.* analyze the distribution of ground truth labels. Most importantly for the following discussion is an identification of nuisance factors [40,44] and robustness criteria that are not explicitly available in image and label space. In the context of automotive and autonomous driving applications, such factors include environmental conditions (*e.g.* rain, dusk) as well as state of the ego-system (*e.g.* view change due to braking maneuver) and other actors in the environment (*e.g.* cyclists and pedestrians).

Hazard Analysis. CV-Hazop [42] is a hazard analysis approach for computer vision. 1470 hazards are currently available for CV-Hazop [1]. A concrete task in our application domain of interest is presented in context of the WildDash dataset [41]. It shows that the extensive list of hazards that are relevant for generic CV tasks can be broken down to a small number of 9 hazard clusters for a concrete segmentation task. Based on CV-Hazop, Zendel *et al.* discuss different aspects of analyzing image data and in particular negative test cases [41]. For negative test cases, we expect the algorithm to fail, yet with expected behavior, *e.g.* signaling high uncertainty. Negative test cases additionally provide a means to check that the limits of algorithms as well as the fidelity of the test environment are well-defined. A similar analysis has been performed by Zhang *et al.* [44] for stereo video focusing on specularity, texturelessness, transparency and disparity jumps. The work also describes mapping of hazards to testing via synthetic data (*cf.* Sect. 2.3). Their analysis shows that algorithms that perform better on average are not necessarily better in handling specified hazards. The Data Safety Guidance [31] describes an approach for data used in safety-related systems and provides concrete guidance such as properties of data that should be considered in the analysis of a data-driven system as well as concrete guide words for a data-focused Hazop analysis similar to CV-Hazop.

Operational Design Domain (ODD). An automated driving system operates in a defined Operational Design Domain [21]. As an example, adverse weather conditions such as heavy rain or snowstorms may be explicitly excluded. The analysis of the ODD is thus very related to the hazard analysis discussed above, as a reduction in ODD may explicitly result in a reduction of relevant hazards. Similarly, an Object and Event Detection and Response (OEDR) describes the proper handling of external situations that the automated vehicle encounters, including perception [21]. The description of ODD and OEDR necessitates an analysis for relevant factors that include similar considerations as described above for CV-Hazop such as weather, glare and sensor noise [21]. However, there are differences between an ODD/OEDR analysis and a CV-Hazop analysis: CV-Hazop is a generic analysis for any CV function, while an ODD/OEDR analysis is focused on a specific AD system implementation. On the one side, due to the generic nature of CV-Hazop, many impact factors need to be considered that may not be relevant for a specific application and the system context the application is embedded in. The result is that we may considerably reduce the number of hazards for a given implementation, because hazards may not apply to the system by design (*cf.* WildDash [41] mentioned above). On the other side, a vision function analyzed based on CV-Hazop may still be directly usable even when the scope in ODD or OEDR are extended.

Top-Down/Bottom-Up. Abstract top-down analysis methods should be complemented with bottom-up data analysis, *e.g.* in an iterative approach: This may include error analysis from machine learning, exploratory data analysis, *e.g.* for confounding factors, coverage considerations, *e.g.* label distribution [8], and novelty detection for interesting tests. As one example, Cordts *et al.* [8] investigate why performance on Cityscapes changes over the seasons and their analysis concluded that this most likely depends on "softer lighting conditions in the frequently cloudy fall". Many issues such as corrupted images or labels, imbalances [8], variations, label noise [11] and preprocessing or data quality issues can only be detected by inspecting the data. While approaches such as data inspections, mis-prediction and outliers analysis are vital, they are rather discussed in practical discussions, *e.g.* [20].

Summary. All discussed analyses support testing by identifying test concerns, hazards, nuisance factors, novel aspects, environmental and operational conditions that should be considered in a test set. For an abstract domain analysis, we need to consider a concretization from abstract domain to tests either with a mapping to a dataset [41] or by generating data synthetically [40,44] as further described below.

Exemplary Research Questions. (*i*) What would be a basic check list of nuisance factors and other hazards that should be considered for a good test set? (*ii*) How do we integrate knowledge from the analysis of the ODD and OEDR into designing a good test set? (*iii*) How to concertize abstract tests into concrete images?

2.3 Synthetic Data

Several works discuss simulation, i.e. 3D rendered graphics, as a key enabler for large-scale testing in the domain of AD [12,22]. Borg *et al.* [4] discuss that a promising approach to ML safety engineering is to simulate test cases. There are several benefits of simulation including (*i*) flexibility and control of the visual effects and the scene content, (*ii*) massive automatic generation of inputs, (*iii*) inherent availability of precise and unambiguous ground truth and (*iv*) early availability in the development cycle.

Synthetic Data Approaches. One specific question for simulation is the required fidelity such that results transfer to the real world [22]. Previous work has mostly focused on comparing average performance. In particular, several authors have reported similar performance between simulations and related datasets [2,24,28,44]. Note that there are several different approaches as to whether (*i*) images are created from learned models (using GANs, *e.g.* DeepRoad [43]), (*ii*) created from a 3D World simulation (*e.g.* for pedestrian detection [28]), (*iii*) augmented with sensor effects such as chromatic aberration and blur [6] and other style transfer, (*iv*) perturbed for robustness testing [27], (*v*) augmented with additional relevant agents and objects (*e.g.* for urban driving scenes [2]) or (*vi*) using probabilistic scene grammars combined with learning an adaption of generated scenes for dataset and task-specific synthetic content generation [19]. The focus of these methods is typically on training and showing a benefit of leveraging synthetic data. Considerations for testing may be different: instead of improving average-case behavior over a realistic distribution, we may rather be interested in improving the least worst-case behavior [10]. Additionally, depending on the type of property, tests may be interested in addressing the content gap and/or the appearance gap of a simulation [19].

Evaluation of Synthetic Data. Note that simulation with a sufficient fidelity has significant cost in creation, maintenance, validation and execution. However, a benefit of simulation is that testing can be scaled economically [40]. Obviously, we can only synthesize what we have already considered and so the focus is on discovering critical interaction of known effects, rather than *unknown unknowns.* A concrete simulation used as a test environment in the context of safety-critical systems needs to consider whether results from simulation transfer to real executions and do not create "false alarms". This concerns many testing tools and has recently been discussed in the context of static analysis results by Meyer [25] from which we reuse the nomenclature. Although from a verification perspective false alarms are not harmful, "false alarms kill an analyzer" [25] w.r.t. user acceptance. Missed violations may be catastrophic and thus need to be specifically considered when using simulation in a safety-related process step. We refer to Meyer [25] for a discussion on the relation of soundness and completeness considerations for verification tools. Note that in the context of testing with synthetic data, we can neither remove all missed violations nor all false alarms and thus need to consider the relative performance. Zhang *et al.* [44] compare evaluation w.r.t. specific hazards (described above) and verify the results on synthetic data with selected, corresponding test data from standard benchmarks, *e.g.* [13]. We refer to Koopman *et al.* [22] for a discussion on residual risk with respect to simulation and its fidelity. These are general simulation concerns, *i.e.* how to accurately represent the real world. One particular difference in the vision domain is that we cannot rely on human evaluation of realism, since two images that may look the same for a human can have a significant difference in algorithmic evaluation in testing [14].

Summary. In general using synthetic data, *e.g.* via simulation, is a commonly-used technique in testing automotive embedded systems. A discussion in the context of simulation models for hardware-in-the-loop systems can be found in [17]. While their focus is on classical automotive systems such as motor control or vehicle stability control, the discussion of model affordances (What sort of functionality should the model provide?) and fidelity (What fidelity does the test need to decide whether a specification is satisfied in the actual system?) is valuable for any simulation-based testing. For the concrete question of fidelity of images and relevant image KPIs in the automotive domain, we also may refer to the *IEEE–SA P2020* working group on automotive imaging standards [15].

Exemplary Research Questions. (*i*) Which affordances should a simulation provide to support building a good test set with domain coverage? (*ii*) How can we leverage synthetic data to economically scale a good test set?

3 Test Evaluation

Labels. Deep learning in computer vision mostly relies on supervised learning, *i.e.* ground truth labels are provided for training. Predictions are compared to ground truth labels based on task-dependent cost metrics such as intersection-over-union (IoU). If synthetic data is used, ground truth labels can be generated automatically. Getting ground truth is a common task for learning, validation and verification and often one of the most expensive and time-consuming tasks [10,19] and a key bottleneck [30]. As an example, fine segmentation maps for Cityscapes required more than 1.5 hours on average [8]. We refer to literature from computer vision [24,40] and benchmarks for autonomous driving [8,13] and machine learning [9,19] for data collection and labeling. There are similarities between manual labeling and manual software testing as both rely on a specification subject to interpretation by humans [10]. One difference is that for V&V we typically want to use high-quality ground truth, while training may also rely on weak signals based on automatic labeling [30]. However, we can compare automatic labeling efforts such as Snorkel [30] with partial specifications in software testing [3] that allow us to automatically test for certain properties.

Properties. While a single cost/loss metric is required for the machine learning problem, a tester is less constrained in the form and number of evaluation functions to apply between ground truth and prediction. Evaluation in machine learning relies on average performance over a dataset, where rare mis-predictions may not have a visible impact. In contrast, the focus of verification is typically on individual tests that each feature a clear pass or fail condition and thus their predictions can be evaluated individually using several complementary criteria. Ground truth is the reference, a point-wise specification, with predictions assumed to be (approximately) equivalent. Ground truth in combination with image augmentation frameworks as described above, can also be used to describe local invariance and equivariance properties: (*i*) *Local invariance*: Small changes on the input do not cause a change on the output. An example is the setup for minimal adversarial examples in image classification as described above, where around images the top-1 classification shall remain the same. (*ii*) *Equivariance*: A change on the input, causes an equivalent change on the output. An example in the context of image augmentation for semantic segmentation is the translation and rotation of a segmentation mask alongside its input image.

Seshia *et al.* [33] survey the landscape of formal specifications for DNNs. The paper mentions that a complete end-to-end perspective may be appealing, since system level constraints, requirements and rules are more intuitive and easier to specify on a system level. Moreover, breaking down constraints and requirements across the functional chain is a challenging task. However, for practical verification a decomposition is necessary.

Metrics. Frtunikj *et al.* [12] discuss a refinement of performance metrics to application-specific quantities depending on safety requirements: This may include the consideration of the environment such as weather and the context of the ego vehicle, *e.g.* in the form of distance- and speed-based mAP (mean average precision). Metrics with more structure are intuitively appealing, since speed and distance are relevant parameters in following stages of the AD functional chain, *e.g.* for computing threat metrics of a planner. The context-dependent performance metrics discussed in [12] are obviously linked to a domain analysis such as CV-Hazop [1]. However, detailed metrics also necessitate availability of the corresponding information for relevant object classes, which is often times not available in standard datasets.

Derived Specifications. There has been work on differential and metamorphic testing of machine learning functions, both so-called derived specifications [3]. First, differential testing or the more general variant of n-version testing is a derived test oracle [3] where the results of n versions of the same function are compared with each other. Here we rely on a comparison of functions that observably differ in their implementations. This can be either based on different implementation approaches or on evolutions of a single implementation, *e.g.* for regression testing. DeepXplore [26] uses differential testing for different applications including a simple example from the AD domain. Note that the use of redundant paths in safety-critical applications, whether through different sensing modalities or redundant implementations, supports a use of differential testing and can help to identify inconsistencies, but not common weak spots. Second, metamorphic testing uses metamorphic relations that check the relative change between different executions of the same SW under test. Note that a metamorphic relation may describe a local (epsilon) invariance property and as such currently formulated adversarial robustness properties are instances of metamorphic relations. Local invariance is also mainly used in previous work on metamorphic testing for classical supervised learning algorithms (classification) on structured data [39]. DeepTest [37] uses the test setup and input generation as DeepXplore, however checking whether a projected driving vector remains invariant within some ϵ bound based on a metamorphic relation. A recent work called DeepRoad [43] uses GANs to apply transformations on the input image, *e.g.* with respect to weather conditions, and checks that the output, in this case driving behavior, stays invariant within some ϵ bound.

Domain Considerations. As in any test setup, if a test execution fails, the error may be due to the implementation or due to the test. For reference-based tests, the actual reference, *i.e.* in our case ground-truth label, may be at fault. There are several sources of errors and label noise [11] including errors of automatic pre-labeling, errors in the labeling specification and errors in labeling [8, 10]. Concrete challenges in labeling are domain-specific. As an example, in AD a fine-grained classification may not be necessary (*cf.* 30 classes in Cityscapes [8]), yet there may be subtle interpretations required, *e.g.* of drivable space on road boundaries. Additionally, in a development environment with separated responsibilities, coordination of labeling specification, the concrete labeling process and quality control of labeling results are important for the high-quality labeling.

Summary. Ground truth is required for verification. For detailed evaluation further task-specific labeling and metadata may be required, especially considering domain-specific evaluation metrics that necessitate details about the environment. However there is an imbalance between the high importance, effort and cost of labeling data in practice versus its prevalence in published works.

Exemplary Research Questions. (*i*) How do we obtain ground truth for diverse test data in a cost-effective manner? (*ii*) What are relevant, task- and domain-specific evaluation metrics? (*iii*) How can we use specifications and formal methods to reap a larger benefit from ground truth data to effectively multiply our test set?

4 Conclusion

We discussed approaches and open research questions for testing of learned CV functions in the particular context of autonomous driving. To this end, we presented existing research from different fields such as software testing, machine learning and computer vision and posed several relevant research questions to provide a basis for a discussion between the individual fields. However, due to space limitations we omitted further topics such as model explainability and interpretability, software, system and ML safety, AD V&V, modeling and simulation, (automatic) labeling, data science and exploratory data analysis.

References

1. AIT Austrian Institute of Technology GmbH: CV-HAZOP VITRO. https://vitro-testing.com/cv-hazop/. Accessed 28 Mar 2019
2. Alhaija, H.A., Mustikovela, S.K., Mescheder, L.M., Geiger, A., Rother, C.: Augmented reality meets computer vision: efficient data generation for urban driving scenes. J. Comput. Vis. **126**(9), 961–972 (2018)
3. Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., Yoo, S.: The oracle problem in software testing: a survey. IEEE Trans. Software Eng. **41**(5), 507–525 (2015)
4. Borg, M., et al.: Safely entering the deep: a review of verification and validation for machine learning and a challenge elicitation in the automotive industry. arXiv preprint arXiv:1812.05389 (2018)
5. Braiek, H.B., Khomh, F.: On testing machine learning programs. CoRR arXiv:abs/1812.02257 (2018)
6. Carlson, A., Skinner, K.A., Vasudevan, R., Johnson-Roberson, M.: Sensor transfer: learning optimal sensor effect image augmentation for sim-to-real domain adaptation. IEEE Robot. Autom. Lett. **4**(3), 2431–2438 (2019)
7. Cheng, C.H., Huang, C.H., Brunner, T., Hashemi, V.: Towards safety verification of direct perception neural networks. arXiv preprint arXiv:1904.04706 (2019)
8. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: CVPR 2016 (2016)
9. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: Imagenet: a large-scale hierarchical image database. In: IEEE CVPR 2009, pp. 248–255 (2009)
10. Franke, U.: 30 years fighting for robustness, June 2018. http://www.robustvision.net. Talk at Robust Vision Challenge (2018)
11. Frénay, B., Kabán, A.: A comprehensive introduction to label noise. In: ESANN 2014 (2014)
12. Frtunikj, J., Fuerst, S.: Engineering safe machine learning for automated driving systems. In: 27th Safety-Critical Systems Symposium (2019)
13. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: IEEE CVPR 2012, pp. 3354–3361 (2012)
14. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
15. Group, W.A.I.Q.W.: Standard for automotive system image quality. In: IEEE P2020, IEEE (2019)
16. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261 (2019)
17. Hutter, A.: Einsatz von Simulationsmodellen beim Test elektronischer Steuergeräte. In: Sax, E. (ed.) Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie. Hanser (2008)
18. Jung, A.: Image augmentation for machine learning experiments (2019). https://github.com/aleju/imgaug. Accessed 9 Apr 2019

19. Kar, A., et al.: Meta-Sim: Learning to Generate Synthetic Datasets. arXiv e-prints arXiv:1904.11621, April 2019
20. Karpathy, A.: A recipe for training neural networks, April 2019. http://karpathy.github.io/2019/04/25/recipe/. blog Accessed 2019 Apr 30
21. Koopman, P., Fratrik, F.: How many operational design domains, objects, and events? In: Workshop on AI Safety @ AAAI 2019 (2019)
22. Koopman, P., Wagner, M.: Toward a framework for highly automated vehicle safety validation. Technical report, SAE Technical Paper (2018)
23. Liu, C., Arnon, T., Lazarus, C., Barrett, C., Kochenderfer, M.J.: Algorithms for verifying deep neural networks. CoRR arXiv:abs/1903.06758 (2019)
24. Mayer, N., et al.: What makes good synthetic training data for learning disparity and optical flow estimation? Int. J. Comput. Vis. **126**(9), 942–960 (2018)
25. Meyer, B.: Soundness and completeness: with precision, April 2019. https://bertrandmeyer.com/2019/04/21/soundness-completeness-precision. blog Accessed 02 May 2019
26. Pei, K., Cao, Y., Yang, J., Jana, S.: DeepXplore: automated whitebox testing of deep learning systems. In: Proceedings of SOSP 2017, pp. 1–18 (2017)
27. Pezzementi, Z., et al.: Putting image manipulations in context: robustness testing for safe perception. In: 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 1–8 (2018)
28. Poibrenski, A., Sprenger, J., Muller, C.: Toward a methodology for training with synthetic data on the example of pedestrian detection in a frame-by-frame semantic segmentation task. In: SEFAIAS@ICSE 2018, pp. 31–34 (2018)
29. Qin, C., et al.: Verification of non-linear specifications for neural networks. CoRR arXiv:abs/1902.09592 (2019)
30. Ré, C.: Software 2.0 and snorkel: beyond hand-labeled data. In: Proceedings of 24th ACM KDD 2018, 19–23 August 2018, p. 2876 (2018)
31. SCSC: Data safety guidance. SCSC Version 3.1, The Safety-Critical Systems Club, York, Great Britain (2019)
32. Sen, K., Marinov, D., Agha, G.: CUTE: a concolic unit testing engine for c. In: ACM SIGSOFT Software Engineering Notes, vol. 30, pp. 263–272. ACM (2005)
33. Seshia, S.A., et al.: Formal specification for deep neural networks. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 20–34. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_2
34. Shetty, R., Schiele, B., Fritz, M.: Not using the car to see the sidewalk: quantifying and controlling the effects of context in classification and segmentation. CoRR arXiv:abs/1812.06707 (2018)
35. Singh, G., Gehr, T., Püschel, M., Vechev, M.T.: An abstract domain for certifying neural networks. PACMPL **3**(POPL), 41:1–41:30 (2019)
36. Sun, Y., Wu, M., Ruan, W., Huang, X., Kwiatkowska, M., Kroening, D.: Concolic testing for deep neural networks. In: Proceedings of ASE 2018, pp. 109–119 (2018)
37. Tian, Y., Pei, K., Jana, S., Ray, B.: DeepTest: automated testing of deep-neural-network-driven autonomous cars. In: arXiv:1708.08559 (2017)
38. Wong, E., Schmidt, F.R., Metzen, J.H., Kolter, J.Z.: Scaling provable adversarial defenses. In: NeurIPS 2018, pp. 8410–8419 (2018)
39. Xie, X., Ho, J.W.K., Murphy, C., Kaiser, G.E., Xu, B., Chen, T.Y.: Testing and validating machine learning classifiers by metamorphic testing. J. Syst. Softw. **84**(4), 544–558 (2011)
40. Yuille, A.L., Liu, C.: Deep nets: What have they ever done for vision? arXiv preprint arXiv:1805.04025 (2018)

41. Zendel, O., Honauer, K., Murschitz, M., Steininger, D., Domínguez, G.F.: Wild-Dash - creating hazard-aware benchmarks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11210, pp. 407–421. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01231-1_25
42. Zendel, O., Murschitz, M., Humenberger, M., Herzner, W.: CV-HAZOP: introducing test data validation for computer vision. In: ICCV 2015, pp. 2066–2074 (2015)
43. Zhang, M., Zhang, Y., Zhang, L., Liu, C., Khurshid, S.: DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In: ASE 2018, pp. 132–142 (2018)
44. Zhang, Y., Qiu, W., Chen, Q., Hu, X., Yuille, A.L.: UnrealStereo: controlling hazardous factors to analyze stereo vision. In: Proceedings of 3DV 2018, pp. 228–237 (2018)