On the Use of Available Testing Methods for Verification & Validation of AI-based Software and Systems

Franz Wotawa
Graz University of Technology, Institute for Software Technology
Inffeldgasse 16b/2, A-8010 Graz, Austria
wotawa@ist.tugraz.at

Abstract

Verification and validation of software and systems is the essential part of the development cycle in order to meet given quality criteria including functional and non-functional requirements. Testing and in particular its automation has been an active research area for decades providing many methods and tools for automating test case generation and execution. Due to the increasing use of AI in software and systems, the question arises whether it is possible to utilize available testing techniques in the context of AI-based systems. In this position paper, we elaborate on testing issues arising when using AI methods for systems, consider the case of different stages of AI, and start investigating on the usefulness of certain testing methods for testing AI. We focus especially on testing at the system level where we are interesting not only in assuring a system to be correctly implemented but also to meet given criteria like not contradicting moral rules, or being dependable. We state that some well-known testing techniques can still be applied providing being tailored to the specific needs.

Introduction

Because of the growing importance of AI methodologies for current and future software and systems, there is a need for coming up with appropriate quality assurance measures. Such measures should come up with certain guarantees that the resulting products fulfill their requirements, e.g., provide the requested functionality and safety concerns. Providing guarantees seem to be essential in order to gain trust in AI-based system solutions. In particular, in autonomous driving to mention one more recent application area of AI, we have to establish a certification and homologation process that assures an autonomous vehicle to follow given regulations and other requirements.

Because of the fact that artifacts making use of AI technology are themselves systems, the question is whether it is possible re-use ordinary testing methodologies and to adapt them for providing means for certification and homologation. In particular, besides components like vision systems relying on machine learning, there are other components that do not rely on any AI methodology. In Figure 1 we give an overview of the architecture of such a system comprising the

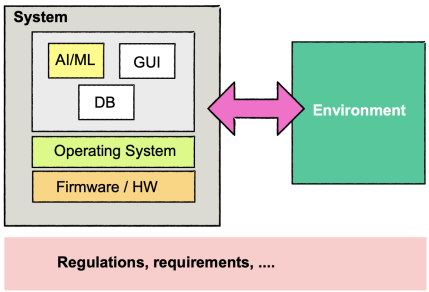

Figure 1: AI-based systems their boundaries, and environment.

AI component and other components implementing functionality like providing user interfaces or database access. In addition, such system rely on a computational stack where we also have to consider the operating system, firmware, and even the hardware for verification and validation purposes. As a consequence, we have to consider verification and validation of the whole system for quality assurance.

In a previous paper (Wotawa 2019), we already focused on the need for system testing. In contrast, in this paper, we try to give a first answer regarding the usefulness of certain available system testing methods for testing AI applications. Furthermore, we discuss the corresponding general verification and validation problem of such application in more detail. We have always to understand what we want to test and what we want to achieve. We have also to be aware of shortcomings arising when focusing only a subparts of the overall verification and validation problem. First, faults often arise because of untested interactions between different system components. Such cases may arise because of unintended interactions not considered during development. Second, we might not be able to sufficiently make guarantees regarding the degree of testing. And finally, we may miss critical inputs or scenarios that lead to trouble. The latter especially holds for different machine learning approaches and is referred to adversarial attacks (see e.g., (Su, Vargas, and Sakurai 2019) and (Goodfellow, McDaniel, and Papernot 2018)).

We organize this paper as follows. In the next section, we

discuss the system testing challenge in detail. We focus on different aspects of testing to be considered and refer to related literature. Afterwards, we present three approaches of systems testing that have been proven to find faults when testing systems using AI techniques. Finally, we summarize the obtained findings.

The testing challenge

As depicted in Figure 1 systems comprising AI methodology also rely on other components providing interfaces and functionality, as well as runtime support including operation systems, firmware, and hardware. As a consequence, we have to consider *testing as a holistic activity* that has to take care of all different parts of the whole system. In particular, we have to *clarify what to test and how to test*. For example, a logic-based reasoning system comprises a compiler for reading in the logic rules and facts, and the reasoning part. Hence, we have to test the compiler and the reasoning part first separately and afterwards together in close interaction. The compiler can be tested, for example using, fuzzing where more or less randomly generated inputs are generated (see e.g., (Köroglu and Wotawa 2019)). The reasoning engine itself can be tested using certain known relations like that the sequence of rules provided to the system does not influence the final outcome (see e.g., (Wotawa 2018)). The overall system itself may be tested using fault injection, e.g., (Wotawa 2016). All these examples have – more or less – in common that they only capture some parts of the expected behavior.

If using fault injection, we are interested in how systems react on inputs that occur in case of faults. When using invariants like the order of rules, we do not test all aspects of reasoning. Hence, in order to thoroughly test such systems, we need to understand what to test in order to *identify shortcomings of underlying testing methods* to be used. Besides this and more specifically to AI methods, we have to provide some measures that at least indicate the quality of testing. For ordinary programs, coverage (e.g., (Ammann, Offutt, and Huang 2003)) and mutation score (e.g., (Jia and Harman 2011)) are used to determine whether test suites are good enough, i.e., being likely able to reveal a faulty behavior. Coverage helps to identify those parts of the program that are executed using the test suite, i.e., code coverage[1]. The mutation score is an indicator of the number of program variants, i.e., the mutations, that can be detected using the given test suite. It is worth noting that coverage or mutation score can be seen as a measure or indicator for guaranteeing that a test suite has the required capabilities for detecting a failing behavior.

Let us consider testing neural networks as an example. Neural networks are trained using a set of examples and evaluated afterwards. Evaluation is used for assuring that a network reaches a given quality of the prediction outcome. The set of examples used for training and evaluation have to be distinct. The question is now whether this evaluation is good enough for replacing further testing effort. The answer is no, because but not only of adversarial attacks (Su, Vargas, and Sakurai 2019) that lead to misclassifications even in case of small input variations. Other reasons for misclassifications are the use of a training data set that is not covering all different examples, and other aspects like the distribution of examples. Furthermore, note that variations of the appearance of objects in the real world exists often. In Figure 2 we depict different images of the traffic sign ”do not enter” ranging from a bend to occlusions because of stickers attached. An autonomous car would require always to handle these case, and it is very unlikely that we really have all of such cases represented in the training data set. Moreover, if so, we still would have misclassifications occurring, requiring to assure that there is no unwanted effect on the behavior of the overall system.

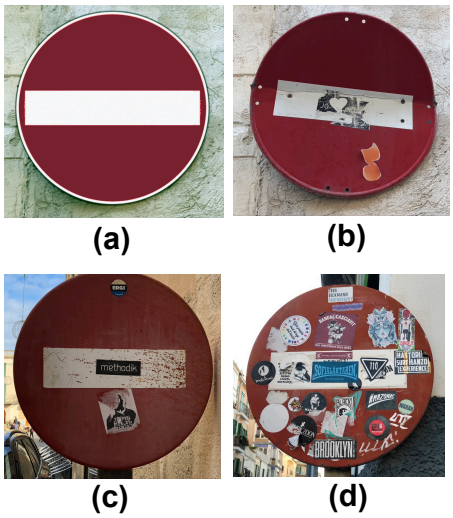

Figure 2: Different variants of the ”do not enter” traffic sign someone sees in reality: (a) is the original sign, (b) the traffic sign with a bend, a sticker, and partially missing color, and (c) and (d) are traffic sign with various stickers attached.

There is plenty of literature regarding different testing approaches for neural networks, e.g., (Pei et al. 2017; Sun, Huang, and Kroening 2018; Ma et al. 2018b,a) and most recently (Kim, Feldt, and Yoo 2019; Sekhon and Fleming 2019). In some of the methods also an adapted version of coverage and mutation score for neural networks has been used. Unfortunately, coverage information maybe somehow misleading (Li et al. 2019) leaving the question regarding the quality of the test suite open.

In the case of neural network we may also ask whether classical coverage or mutation score used in ordinary software engineering can be used as quality measure when testing a current neural network implementation. (Chetouane, Klampfl, and Wotawa 2019) showed that making use of these measures when testing the configuration of neural networks, i.e., setting the type of neurons, the number of layers and neurons, can be justified. Unfortunately, this is not the case when testing the whole neural network library as discussed in (Klampfl, Chetouane, and Wotawa 2020). Hence, for neural networks or measures and means for testing shall

[1]Note that besides code coverage there are other coverage definitions used like test input coverage, combinatorial coverage, etc.

be provided.

Although, we may need to live with the challenge that we cannot completely tests certain system parts and that there is always a critical case where the AI part of a system may deliver a wrong result, the further question is whether this establishes a problem for the whole system. The answer in this case is no, providing that the system itself is able to detect this critical case and to react appropriately. For example in autonomous driving, we may make use of more than one sensor for obtaining information regarding objects around the vehicle and use sensor fusion to obtain reliable information. We only need to assure that the whole system interacts with the environment in a way that is dependable and fulfills our requirements including ethical or moral considerations. Hence, identifying critical scenarios between the system and its environment seems to be a crucial factor of testing AI-based systems (Koopman and Wagner 2016; Menzel, Bagschik, and Maurer 2018).

Moreover, it seems also of importance to consider that critical scenarios often originate from different settings that have to occur at the same time. One issue, e.g., missing a certain traffic sign may not lead to an accident, but in combination with other issues would.

We summarize our discussion in the following position:

Position 1 *Testing aims at identifying interactions between the system under test and its environment leading to an unexpected behavior. When testing systems utilizing AI, we have to consider testing all parts of a system including the one with and the one without AI as well as their interactions. Evaluating performance characteristic of implemented AI methodology may not be sufficient for assuring meeting quality criteria.*

Most of testing is performed during development of systems before deployment. In some cases certification (or even homologation), i.e., the formal confirmation that an application, product or system, meets its required characteristics, is needed. In case of AI technology we are interested that the system fulfills dependability goals like safety but maybe also given ethical or moral rules. For example, we want a conversational agent or a decision support system not to be racist or sexist. Furthermore, because of the fact that the system's underlying software is updated regularly in order to cope with changes required because of bugs or improved functionality, there is a need of carrying out any certification regularly as well. For example, in autonomous driving we have to assure that a new software update is not going to lead to an unsafe system. However, regression testing may require a lot of effort or come with high costs, which may be reduced when automating testing.

Hence, automating at least part of certification may be a future requirement. But how can certification of AI be carried out? What we need is a process where we identify what we want to achieve, and how this can be checked (or tested)? How can we come up with certain parameters justifying that testing is appropriate? We shall also think about the methods for checking, their limitations, and how to assure that the methods can guarantee (with respect to a given certainty) that the system fulfills requested needs. However, in any case in order to bring AI technology into practice, we have to convince customers that the systems are not of harm. Certification that takes into account such customer's considerations as well as regulations provide the right means for further supporting the delivery of AI technology into practical applications we are using on a daily basis.

It is worth noting that there are many initiatives like the ethics guidelines for trustworthy AI (Pietilä et al. 2019) for coming up with first steps of how AI-based systems have to be constructed, evaluated, and verified. However, for example, in autonomous driving such principles have to be concretized leading to practical rules companies can follow when developing AI systems or systems at least partially based on AI methodologies and tools.

Position 2 *There is a need for well-defined certification and homologation processes for AI-based systems that ideally can be carried out in an automated way. Such certification and homologation processes shall rely on existing guidelines considering all aspects of trustworthy AI.*

When we want to carry out certification at least partially automated we may rely on testing. Hence, we have to state the question whether existing testing techniques can be used for confirming that an AI-based system fulfills regulation and other rules and expectations. This includes besides testing functionality the degree of fulfilling generally agreed ethical and moral rules. In the following section, we introduce three techniques that can (at least partially) serve this purpose.

Testing AI

As discussed there seems to be a need for testing the whole system considering functional and non-functional requirements including moral and ethical rules. For testing systems at the system level black-box approaches are used that do not consider the internal structure. Various methods with corresponding tools have been proposed including model-based testing (MBT) (Utting and Legeard 2006), combinatorial testing (CT) (Kuhn et al. 2015), or metamorphic testing (Chen, Cheung, and Yiu 1998). MBT makes use of a model of the system for obtaining test cases. In order to find critical interactions between the system and its environment this may not be sufficient. It would be required to model the environment including potential interactions and have a look about the reactions of the system.

The focus on modeling the environment of the system in order to obtain test cases is somehow different to ordinary MBT where a model of the system is used for test case generation. Changing from modeling the system to modeling the environment is necessary for finding critical interactions between an AI-based system and its environment. Moreover, in this kind of testing we are not interested in showing that an implementation works accordingly to a model, but is capable of handling arbitrary interactions that may not be foreseen during development.

In contrast to MBT, CT has been developed to search for critical interactions between configuration parameters and inputs. It has been shown that CT can effectively detect faults in many different kinds of software (Kuhn et al. 2009).

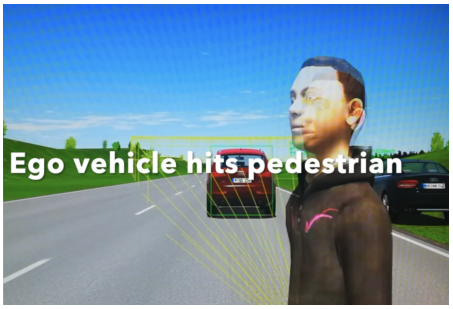

Figure 3: The last episode of a failing test case applied to an implementation of an automated emergency braking system, close to the time where a simulated pedestrian tries to cross the street coming from the right side. The crash occurred in a scenario where another vehicle at the front brakes, causing the ego vehicle to brake. A first pedestrian crossing the street from left passing by, and the second one coming from right who is overseen by the automated emergency braking system and hit.

The question is whether we can also apply CT for AI testing? In (Li, Tao, and Wotawa 2020) the authors introduced an approach utilizing a model of the system environment in combination with CT for obtaining a test suite. In their paper, the authors not only provide the foundations but also reported on a case study where the authors tested an automated emergency braking (AEB) function. From 319 test cases, 9 test cases lead to crashes (including test cases where pedestrians would have been killed (see Figure 3)), and 30 were considered as being critical. It is worth noting that the proposed overall approach also includes a simulation environment for carrying out the generated test cases in a realistic setting automatically.

(Klück et al. 2019) introduced an alternative method for generating critical scenarios, where the authors rely on genetic algorithms for obtaining test cases. The idea is to model test cases as genes that can be crossed and mutated. The evaluation function maps test cases to a goodness value. In each generation the best test cases are taken modified and again evaluated. This kind of testing is also referred to search-based testing. In (Klück et al. 2019) the authors also evaluated the approach using an AEB function too. The obtained results showed that genetic algorithms can be applied to detect faults in the setting of autonomous and automated driving leading to the following position:

Position 3 *Combinatorial testing and search-based testing are effective testing techniques for identifying critical scenarios.*

CT and also search-based testing applied to test autonomous and automated driving functions always has to fulfill the property that no crash with another car or even a pedestrian occurs. In this context closeness to a crash is often represented as the time to collision (TTC), where 0 means that a crash occurs. Usually, in many applications positive but small TTC values may also be considered as unwanted. When testing in the case of the automotive domain including autonomous driving, we can always rely on the TTC for judging whether a test case passes or fails.Hence, there the test oracle can be automated using the TTC, which is not always the case when testing AI. We, therefore, require other means for dealing with the oracle problem, i.e., providing a function that allows to distinguish passing executions of programs and systems from failing ones.

The objective behind metamorphic testing (Chen, Cheung, and Yiu 1998) is to provide a solution to the oracle problem of testing. The underlying idea is to define relations over different inputs that always deliver the same output. For example, $\sin(x)$ is equivalent for all values of x and $x + 2 \cdot \pi$, i.e., $\sin(x) = \sin(x + 2 \cdot \pi)$ always holds. In (Guichard et al. 2019) and more specifically in (Bozic and Wotawa 2019) the authors proposed the use of metamorphic testing for testing conversational agents, i.e., chatbots. The underlying described idea was to propose relations considering semantical relationships between words and sentences, e.g., some sentences have the same semantics when replacing one word with its synonym, or sometimes the sequence of sentences given to a chatbot, does not change the answer provided by the chatbot. Moreover, we are able to test for fulfilling certain moral and ethical regulations. For example, if an answer of a chatbot should not be influenced by the race or sex of the chat participant, we can be formulate this as a metamorphic relation, where we say that a conversion considering one race or sex should lead to the same results when changing race or sex. In case of AI systems, where we are able to come up with metamorphic relations, we are also able to apply metamorphic testing for solving the oracle problem.

Position 4 *Metamorphic testing seems to be of use for implementing the test oracle problem of AI systems allowing to identifying contradictions with requirements, which may include ethical and moral considerations.*

There are more system testing approaches that can be also adapted to fit the purpose of AI testing with the objective of assuring safety of AI-based systems and software. However, we have identified approaches where there is experimental evidence that they could be effectively used for testing AI-based systems. These approaches may also fit into certification and homologation processes. For this purpose, certain measures have to be developed that can be used for deciding when to stop testing in cases no failing test case could be obtained.

Moreover, the presented methods and techniques for testing AI-based systems have disadvantages. They are mainly focusing on quality assurance of the overall system and not its comprising parts. For example, the CT approach considers a model of the environment, which works as basis for obtaining the CT input model. The approach is testing whether certain interactions of the CT with its environment reveal a fault, and in case of automated or autonomous driving a crash, but does not consider any knowledge regarding the SUT's internal structure or behavior. Finding out the root cause of any misbehavior within the SUT might be complicated. Moreover, we are not able to make use of quality assurance measures like code coverage or mutation score for

the particular test suite. Furthermore, CT like MBT requires to concretize the abstract test cases computed using these testing methods. This concretization step cause additional effort and has to be done carefully in order to come up with good test cases that can be executed and most likely reveal a fault.

In case of metamorphic testing it is essential to define the metamorphic relations, which cause additional effort and influence the ability to work as a good test oracle. There are maybe metamorphic relations leading to test cases a SUT can easy fulfill only allowing to test a fraction of functionality. In such cases metamorphic testing would not lead to tests covering most of the functionality and, therefore, can be considered as incomplete. Search-based testing requires to implement a search procedure using a function allowing to estimate the quality of a current test, e.g., the ability of a test revealing a fault. Again this requires additional effort and costs. It is worth noting that in some cases random testing, i.e., generating test inputs using a random procedure, also provides fault revealing test cases requiring even less time than search-based testing at almost no additional costs.

Conclusion

In this position paper, we focused on providing an answer to the question whether there exists testing techniques that can be efficiently used for checking that a software or system comprising AI methodologies fulfills requirements including also moral and ethical rules, and regulations. We also discussed the involved challenges of testing where we identified also shortcomings that arise when only focusing on specific parts and not providing a holistic view. Finally, we introduced several testing methods that have been developed in the context of testing ordinary systems and elaborate on their usefulness in the context of AI-based systems. Search-based testing, combinatorial testing, and metamorphic testing seem to be excellent candidate for this purpose and may also be of use for automating certification and homologation processes for AI applications.

However, further studies have to be carried out. For CT more experiments making use of other autonomous and automated functions have to be considered. Moreover, we require to come up with certain measures of guarantees for the computed test suites. Parameters of CT like the combinatorial strength maybe sufficient but in the context of AI-based systems there is no experimental evidence. For metamorphic testing we further need more use cases and experimental evaluations making use of AI-based systems. In the case of chatbots and also logic-based reasoning metamorphic testing has already been successfully applied. However, there is a need to show the usefulness of metamorphic testing also in other applications where AI technology is a central part.

Acknowledgments

The research was supported by ECSEL JU under the project H2020 826060 AI4DI - Artificial Intelligence for Digitising Industry. AI4DI is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2019 and April 2022. More information can be retrieved from https://iktderzukunft.at/en/ bmvit.

References

Ammann, P.; Offutt, J.; and Huang, H. 2003. Coverage Criteria for Logical Expressions. In *Proceedings of the 14th International Symposium on Software Reliability Engineering*, ISSRE '03. Washington, DC, USA: IEEE Computer Society.

Bozic, J.; and Wotawa, F. 2019. Testing Chatbots Using Metamorphic Relations. In Gaston, C.; Kosmatov, N.; and Le Gall, P., eds., *Testing Software and Systems*, 41–55. Cham: Springer International Publishing. ISBN 978-3-030-31280-0.

Chen, T.; Cheung, S.; and Yiu, S. 1998. Metamorphic testing: a new approach for generating next test cases. Technical report, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong. Technical Report HKUST-CS98-01.

Chetouane, N.; Klampfl, L.; and Wotawa, F. 2019. Investigating the Effectiveness of Mutation Testing Tools in the Context of Deep Neural Networks. In *IWANN (1)*, volume 11506 of *Lecture Notes in Computer Science*, 766–777. Springer.

Goodfellow, I.; McDaniel, P.; and Papernot, N. 2018. Making Machine Learning Robust Against Adversarial Inputs. *Commun. ACM* 61(7): 56–66. ISSN 0001-0782. doi:10.1145/3134599. URL http://doi.acm.org/10.1145/3134599.

Guichard, J.; Ruane, E.; Smith, R.; Bean, D.; and Ventresque, A. 2019. Assessing the Robustness of Conversational Agents using Paraphrases. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 55–62.

Jia, Y.; and Harman, M. 2011. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering* 37(5): 649–678.

Kim, J.; Feldt, R.; and Yoo, S. 2019. Guiding Deep Learning System Testing Using Surprise Adequacy. In *Proceedings of the 41st International Conference on Software Engineering*, ICSE'19, 1039–1049. IEEE Press. doi:10.1109/ICSE.2019.00108. URL https://doi.org/10.1109/ICSE.2019.00108.

Klampfl, L.; Chetouane, N.; and Wotawa, F. 2020. Mutation Testing for Artificial Neural Networks: An Empirical Evaluation. In *IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, 356–365. IEEE.

Klück, F.; Zimmermann, M.; Wotawa, F.; and Nica, M. 2019. Performance Comparison of Two Search-Based Testing Strategies for ADAS System Validation. In Gaston, C.; Kosmatov, N.; and Le Gall, P., eds., *Testing Software and Systems*, 140–156. Cham: Springer International Publishing. ISBN 978-3-030-31280-0.

Koopman, P.; and Wagner, M. 2016. Challenges in Autonomous Vehicle Testing and Validation. *SAE Int. J. Trans. Safety* 4: 15–24. doi:10.4271/2016-01-0128. URL https://doi.org/10.4271/2016-01-0128.

Köroglu, Y.; and Wotawa, F. 2019. Fully automated compiler testing of a reasoning engine via mutated grammar fuzzing. In Choi, B.; Escalona, M. J.; and Herzig, K., eds., *Proceedings of the 14th International Workshop on Automation of Software Test, AST@ICSE 2019, May 27, 2019, Montreal, QC, Canada*, 28–34. IEEE / ACM. doi:10.1109/AST.2019.00010. URL https://doi.org/10.1109/AST.2019.00010.

Kuhn, D.; Kacker, R.; Lei, Y.; and Hunter, J. 2009. Combinatorial Software Testing. *Computer* 94–96.

Kuhn, D. R.; Bryce, R.; Duan, F.; Ghandehari, L. S.; Lei, Y.; and Kacker, R. N. 2015. Combinatorial Testing: Theory and Practice. In *Advances in Computers*, volume 99, 1–66.

Li, Y.; Tao, J.; and Wotawa, F. 2020. Ontology-based test generation for automated and autonomous driving functions. *Inf. Softw. Technol.* 117. doi:10.1016/j.infsof.2019.106200. URL https://doi.org/10.1016/j.infsof.2019.106200.

Li, Z.; Ma, X.; Xu, C.; and Cao, C. 2019. Structural Coverage Criteria for Neural Networks Could Be Misleading. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, 89–92.

Ma, L.; Zhang, F.; Sun, J.; Xue, M.; Li, B.; Juefei-Xu, F.; Xie, C.; Li, L.; Liu, Y.; Zhao, J.; et al. 2018a. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, 100–111. IEEE.

Ma, L.; Zhang, F.; Xue, M.; Li, B.; Liu, Y.; Zhao, J.; and Wang, Y. 2018b. Combinatorial testing for deep learning systems. *arXiv preprint arXiv:1806.07723* .

Menzel, T.; Bagschik, G.; and Maurer, M. 2018. Scenarios for Development, Test and Validation of Automated Vehicles. In *arXiv:1801.08598*. URL https://arxiv.org/abs/1801.08598. Appeared in Proc. of the IEEE Intelligent Vehicles Symposium.

Pei, K.; Cao, Y.; Yang, J.; and Jana, S. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, 1–18. ACM.

Pietilä, P. A.; et al. 2019. Ethics Guidelines For Trustworthy AI. High-Level Expert Group on AI, European Commission.

Sekhon, J.; and Fleming, C. 2019. Towards Improved Testing For Deep Learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, 85–88.

Su, J.; Vargas, D. V.; and Sakurai, K. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* 1–1. ISSN 1089-778X. doi:10.1109/TEVC.2019.2890858.

Sun, Y.; Huang, X.; and Kroening, D. 2018. Testing deep neural networks. *arXiv preprint arXiv:1803.04792* .

Utting, M.; and Legeard, B. 2006. *Practical Model-Based Testing - A Tools Approach*. Morgan Kaufmann Publishers Inc.

Wotawa, F. 2016. Testing Self-Adaptive Systems using Fault Injection and Combinatorial Testing. In *Proceedings of the Intl. Workshop on Verification and Validation of Adaptive Systems (VVASS 2016)*. Vienna, Austria.

Wotawa, F. 2018. Combining Combinatorial Testing and Metamorphic Testing for Testing a Logic-based Non-Monotonic Reasoning System. In *In Proceedings of the 7th International Workshop on Combinatorial Testing (IWCT) / ICST 2018*.

Wotawa, F. 2019. On the importance of system testing for assuring safety of AI systems. In *CEUR Workshop Proceedings , Workshop on Artificial Intelligence Safety, AISafety 2019*, volume 2419. Macao, China. URL http://ceur-ws.org/Vol-2419/.