https://doi.org/10.20378/irb-47275

A Survey on Methods for the Safety Assurance of Machine Learning Based Systems

Gesina Schwalbe, Martin Schels

A Survey on Methods for the Safety Assurance of Machine Learning Based Systems

Gesina Schwalbe and Martin Schels

Artificial Intelligence and Robotics Laboratory,
Continental AG, Regensburg, Germany
{forename.lastname}@continental-corporation.com

Abstract. Methods for safety assurance suggested by the ISO 26262 automotive functional safety standard are not sufficient for applications based on machine learning (ML). We provide a structured, certification oriented overview on available methods supporting the safety argumentation of a ML based system. It is sorted into life-cycle phases, and maturity of the approach as well as applicability to different ML types are collected. From this we deduce current open challenges: powerful solvers, inclusion of expert knowledge, validation of data representativity and model diversity, and model introspection with provable guarantees.

Keywords: functional safety, life-cycle, ISO 26262, machine learning, explainable AI

1 Introduction

Machine learning is currently rapidly finding its way into the domain of safety critical applications in the scope of the ISO 26262 automotive functional safety standard. Examples are computer vision systems or trajectory planning for autonomous driving (AD) [25, 39]. The reason for this advance of machine learned models is its applicability to inherently complex, little understood problems merely via sample data. Unfortunately, this benefit is often accompanied by a mostly black-box character and high complexity of the final model in use, rendering conventional methods for safety assurance insufficient or inapplicable. For example, inductive/deductive inspection and extensive interface/unit testing are inapplicable due to model complexity. The same holds for the "proven in use" argument due to the problem complexity. The latter is reasoned in [21], which historically derives problems for risk assessment of AI. A mathematical deduction can be found in [39, sec. 2.2].

In [37], a complete applicability assessment of ISO 26262 methods to deep neural networks (DNNs) was conducted. They also found that design guidelines, model inspection techniques, formal verification and training data assessment techniques are required. Such new methods for verification and validation were loosely collected e.g. in the surveys [44, p. 36–37] and [19], to which we would like to refer the reader for a more extensive method catalogue. In this paper we proceed with the next step: Sort available methods (respectively method categories) into the main stages of the standard development life-cycle, and from this systematically identify open challenges. This paper provides:

- a *short overview* of ML specific approaches and methods supporting a safety case, *structured* by the phases of the life-cycle (see Tab. 1 for a summary). The focus will be on DNNs as in [37] in the context of AD.
- evaluation of their applicability to different ML types.
- identification of *open challenges* for safety argumentation.

We claim that this novel view on ML development methodologies from a certification perspective is a key to enable ML in safety critical applications. The view should be seen as a top-down approach towards a safety argumentation strategy.

2 ML Specific Safety Argumentation

There are important differences between the traditional approach to the development and certification of safety critical software and the data driven paradigm of machine learning. This is addressed in the following, aligned to a typical life-cycle. The details of the methods, their applicability and open challenges are summarized in Table 1.

Our life-cycle skeleton is based on the ISO 26262 part 6 process [1, Figure 1] for the development of automotive software-enabled components. The main idea of the V-model suggested there is a feedback loop consisting of the phases requirements gathering, (model) development, and finally verification and validation (and integration, which is not considered here). During *requirements engineering* the system architecture is constructed up to smallest functional units, and for each the use-case is specified, including intended operational environment and behavior. Both are guided by verifiable safety goals identified in an inductive hazard and risk analysis. In the succeeding *development* phase, the model design and implementation on hardware and software side is conducted. This is succeeded by intense *verification* and *validation* based on the given requirements. The results are used to determine, whether the model (verification) or the requirements (validation) need to be updated.

In ML, the model design is data driven and automatized, and the implementation consists of classically implemented interpreters for the generated model definitions. We here solely concentrate on the new aspects introduced by the design, verification and validation of (ML) models.

2.1 Requirements Engineering

Available expert knowledge and experience should be compiled into the use-case, system and function requirements formulation, and best be formulated in specialized KPIs. With data driven development of (possibly non-robust) black boxes it becomes practically hard to do this for the little available domain knowledge. As categories of safety requirements for NNs we identified: data representativity, such as

- **scenario coverage** (for further metric suggestions see collection in [8]) applied to an input space ontology such as developed in [5] for the context of autonomous driving;
- **robustness** of the algorithm, such as adversarial robustness [e.g. definition in 22, sec. 6] of NNs;
- **fault tolerance** of the system, as could be increased by runtime monitoring and traditional model redundancy [2, (E) 7.4.12] which requires a model diversity measure;
- **safety related performance** requirements for the black-box, i.e. specialized performance measures such as detection performance for occluded objects [8];
- **plausibility** of the algorithm behavior, such as sensible intermediate steps, or modeling quality of domain specific physical rules, like gravity, translation invariance, or legal ones like collected for the definition of responsibility sensitive safety for AD trajectory planning in [39]; and finally
- requirements arising from **experience** about algorithm specific faults, such as can be found in the AD specific collection of problems and faults in [28].

These cornerstones need to be respected during requirements gathering. Data representativity and model diversity measures are open challenges since methods are scarce and untested.

2.2 Development

Prevention of faults by proper design choices during development are an important building block for a safety case. Some ML specific quality criteria are collected below. Since meaningful KPIs are necessary in order to conduct quantitative analyses, suggestions on such are given as well.

Design based on experience All general design choices not specified in the requirements need to be reasoned, best by experts in the field. Choices include at least the training objective, the model type, the training method including early stopping criteria, the micro design—e.g. activation functions, topology, and loss function for NNs—and initialization values, for both hyperparameters and parameters meant for optimization. An example collection of five design problems for the machine learning area of reinforcement learning with a focus on NN solutions can be found in [4].

Incorporation of uncertainty Most machine learning models will not provide a proper uncertainty output, i.e. estimation of the prediction variance. [29] advises to propagate uncertainty of any component through the system to enable monitoring of accumulated uncertainty. Examples of ML models with uncertainty output are classical Bayesian networks, or for NNs Bayesian new deep learning methods like treating weights as distributions [14], specialized loss (e.g. [38] models classification output as Dirichlet distributions), or via learned specialized weight decay and dropout [23]. Uncertainty treatment was also shown to improve properties like adversarial robustness [38]. Even though available methods are theoretically mature, there is little practical experience and a lot of ongoing research in this direction.

Inclusion of expert knowledge Available expert knowledge should be included into the model in a measurable way for better behavior control. Ways to include prior knowledge are e.g. via the training data as shown in [9], which is framework for automated counterexample generation and training data augmentation; via the loss function as done for soft logic in a teacher-network approach in [18], for fuzzy logic in [35], and for a general objective function in [13]; via statistical model repair methods for reinforcement learning (RL) as developed in [15]; via safe state enforcement for RL as can be achieved through the adaptive control strategy developed in [3] and through the planner switching strategy suggested in [12]; or via topology as in convolutional NNs (CNNs) or in the ReNN architecture [42] which introduces an interpretable and verifiable intermediate layer. All mentioned methods report to also increase performance, but yield little guarantees and are an open field of research.

Robustness Enhancements Robustness here means the indifference of the model output for slight (with regard to a given metric) changes for the given data samples. Non-robust models exhibit chaotic behavior, and deprive testing data of their significance since one sample generalizes to a comparatively small input range. Furthermore, measures must be taken to increase and assure robustness, as e.g. regularization and training data preparation, either by adding adversarial counterexamples [9] or by removing non-robust features in the data [20]. Uncertainty treatment was shown to help as well (see above).

Proper design choices during development contribute to a convincing process argumentation. However, for data-driven automated modeling combined with high complexity, process argumentation needs to be supported by a strong product based argumentation based on verification and validation.

2.3 Verification

Verification is the formal check of the model against the defined (formal) requirements and testing data. This requires tools (e.g. solver or search algorithms) to conduct the formal model checks on the novel ML models. Means to check properties in form of rules are:

- **solvers**, e.g. based on satisfiability modulo theory (SMT) [see survey in 6], or mixed integer linear programming [10], where for both approaches the idea is to transform the condition on the network to check into a solvable equation system;
- **output bound estimation** such as ReluVal [43] optimized for networks with ReLU activations, or the more general reachability analysis tool DeepGo [36];
- **search algorithms** like [31].

The given approaches are well matured. However, they specialize on stability properties, except for the very performance limited solvers. Therefore, we see a need for rule checkers that can efficiently deal with first-order logic like the SMT based solver Reluplex [22].

2.4 Validation

Validation is the task of identifying and adding missing requirements and test cases. It takes up an essential role in a ML safety case due to the sparse specification typical for ML tasks. We identified two aspects that need to be assessed: The testing and training data, and the inner logic and representation of the ML model. Methods for the latter can be categorized in qualitative and quantitative.

Data validation Test and training cases should thoroughly cover the input space. Test cases additionally need to cover the available experience, and the model behavior. The first two were described above. For model behavior coverage, new NN specific metrics were suggested alongside a test case generation framework in [40], and mature tools for counterexample generation [9] are available such as [16] which maximizes the model output differences for minimal perturbations using efficient fuzzy testing.

Qualitative analysis Qualitative analysis may give experts valuable indications about misbehavior. One type is the assessment of "attention", i.e. the assessment which parts of the input most contributed to an output decision. In the case of computer vision, heatmapping can be used to indicate attention. There are both model specific methods available like the signal back-tracing methods suggested in [27],as well as model agnostic ones like based on local linear approximations [32, 34]or on mutual information like the attention estimator trained in [7]. Local linear approximations are created by observing the output for slight, spacially distinct, modifications of one given example. Such modifications can be blending out parts like in the original LIME method [34] or e.g. more advanced blurring techniques as in RISE [32]. Other than attention analysis, feature visualization [30] for NNs gives insight into the specialization of single units. Additional explanatory output like textual explanations as demonstrated in [26] can also be useful. Another example of additional output is hierarchical information as realized in [35] by additional output neurons together with logical constraints translated into the loss function.

Quantitative analysis Methods that yield quantitative insights into black-box NNs are scarce, especially for NNs. One is extraction of rules, either locally for a subset of examples using inductive logic programming [33] or globally. Global model-agnostic rule extraction like validity interval analysis [41], which iteratively refines valid intervals using back-propagation of outputs, is due to the complexity of concurrent networks not applicable to NNs (see the survey on rule extraction for NNs in [17]). Other methods are analysis of the performance on sub-tasks [13] and of the inherent (distributed) representation of semantic domain concepts within the network structure [11, 24]. The latter learn the representation of a concept within the NN from its intermediate output on a training set for this concept. Net2Vec [11] does this by attaching an additional output, which enables retraining for the concept, while TCAV [24] uses support vector machines to improve uniqueness. Given a representation vector of a semantic concept, its attribution to output classes can be determined via directed derivative along the found representation vector [24]. Concept analysis could enable topological modularization as in the simple example in the ReNN architecture [42].

3 Conclusion and Outlook

ML requires new approaches for a convincing safety argument, and such are available for different ML types. The identification of open challenges is eased by a structured, certification oriented overview of these methods such as summarized from the above in Table 1. This now needs to be extended and aligned with a safety argumentation strategy, to finally be established in standards and industries.

Table 1. Overview of method categories for the safety argumentation of ML models; for each category examples are given, accompanied by citations for further reading and by the applicability scope restrictions (applicable to all ML models if none are given). The methods are categorized by the life-cycle phase they are applicable in. Major open challenges are marked *italic*.

Phase & method goals	Method categories	Examples	Scope
Requirements Engineering Use available domain knowledge to formulate use-case and safety requirements	*Data representativity requirements*	- scenario coverage [8] - input space ontology [5]	AD
	Robustness requirements	- adversarial robustness metric [22]	
	System fault tolerance requirements	- runtime monitoring - *model diversity measure* for redundancy [2, 7.4.12]	
	Safety performance measures	- occlusion sensitivity [8]	
	Plausibility requirements	- AD behavior rules [39] - sensible intermediate steps - domain specific rules (physical, legal)	TP[a]
	Experience collection	- AD pitfalls collected in [28]	AD
Development Apply reasonable design choices at all decision points	Design based on experience	- experience on model type, training method, initialization values - list of pitfalls in [4]	RL
	Incorporation of uncertainty [29]	- [14, 23]	NNs
	Inclusion of expert knowledge	- via loss function [18, 35] - via topology [42] - model repair [15] - safe learning [3, 12]	NNs NNs RL RL
	Robustness enhancements	- regularization [see 20] - robustification of training data [20] - counterexample-guided data augmentation [9]	
Verification Check against test data and model requirements	Formal verification of rules, model, KPIs	- *solvers* [6, 10] - boundary approximation [36, 43] - search algorithms [31]	NNs NNs NNs
Validation Find missing . . .			
. . . test cases	Input space coverage checks	- (see data representativity method)	
	Experience coverage checks	- (see experience collection method)	
	Model coverage checks	- concolic testing [40] - counterexample generation [9, 16, 40]	NNs NNs
. . . requirements via qualitative model analysis	Attention analysis through heatmapping [45]	- back-propagation based, e.g. [27] - model agnostic, e.g. [7, 32, 34]	NNs
	Feature visualization	- method collection in [30]	CNNs
	Explanatory output	- textual explanations [26] - hierarchical information [35]	NNs NNs
. . . requirements via *quantitative model analysis*	Rule extraction	- locally via ILP [33] - global model agnostic, e.g. VIA [41] - global NN specific methods [17]	 NNs
	Sub-task/concept analysis	- Neural Stethoscopes [13] - concept embedding and attribution analysis [11, 24] - ReNN modularized topology [42]	NNs NNs NNs

[a] trajectory planning

Bibliography

[1] 32, I.S.: Road Vehicles — Functional Safety — Part 1: Vocabulary, vol. 1, 2 edn. (2018)

[2] 32, I.S.: Road Vehicles — Functional Safety — Part 6: Product Development at the Software Level, vol. 6, 2 edn. (2018)

[3] Akametalu, A.K., Fisac, J.F., Gillula, J.H., Kaynama, S., Zeilinger, M.N., Tomlin, C.J.: Reachability-based safe learning with gaussian processes. In: Proc. 53rd IEEE Conf. Decision and Control, pp. 1424–1431 (2014)

[4] Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., Mané, D.: Concrete problems in AI safety. CoRR **abs/1606.06565** (2016)

[5] Bagschik, G., Menzel, T., Maurer, M.: Ontology based scene creation for the development of automated vehicles. In: Proc. 2018 IEEE Intelligent Vehicles Symp., pp. 1813–1820 (2018)

[6] Bunel, R.R., Turkaslan, I., Torr, P., Kohli, P., Mudigonda, P.K.: A unified view of piecewise linear neural network verification. In: Advances in Neural Information Processing Systems 31, pp. 4790–4799 (2018)

[7] Chen, J., Song, L., Wainwright, M., Jordan, M.: Learning to explain: An information-theoretic perspective on model interpretation. In: Proc. 35th Int. Conf. Machine Learning, vol. 80, pp. 883–892 (2018)

[8] Cheng, C.H., Nührenberg, G., Huang, C.H., Ruess, H., Yasuoka, H.: Towards dependability metrics for neural networks. In: 16th ACM/IEEE Int. Conf. Formal Methods and Models for System Design, pp. 43–46 (2018)

[9] Dreossi, T., Ghosh, S., Yue, X., Keutzer, K., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Counterexample-guided data augmentation. In: Proc. 27th Int. Joint Conf. Artificial Intelligence, pp. 2071–2078 (2018)

[10] Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks. In: Proc. 10th Int. Symp. NASA Formal Methods, vol. 10811, pp. 121–138 (2018)

[11] Fong, R., Vedaldi, A.: Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In: Proc. 2018 IEEE Conf. Comput. Vision and Pattern Recognition, pp. 8730–8738 (2018)

[12] Fridovich-Keil, D., Herbert, S.L., Fisac, J.F., Deglurkar, S., Tomlin, C.J.: Planning, fast and slow: A framework for adaptive real-time safe trajectory planning. In: Proc. 2018 IEEE Int. Conf. Robotics and Automation, pp. 387–394 (2018)

[13] Fuchs, F.B., Groth, O., Kosiorek, A.R., Bewley, A., Wulfmeier, M., Vedaldi, A., Posner, I.: Neural Stethoscopes: Unifying analytic, auxiliary and adversarial network probing. CoRR **abs/1806.05502** (2018)

[14] Gast, J., Roth, S.: Lightweight probabilistic deep networks. In: Proc. 2018 IEEE Conf. Comput. Vision and Pattern Recognition, pp. 3369–3378 (2018)

[15] Ghosh, S., Lincoln, P., Tiwari, A., Zhu, X.: Trusted machine learning: Model repair and data repair for probabilistic models. In: Workshops 31st AAAI Conf. Artificial Intelligence, vol. WS-17 (2017)

[16] Guo, J., Jiang, Y., Zhao, Y., Chen, Q., Sun, J.: DLFuzz: Differential fuzzing testing of deep learning systems. In: Proc. ACM Joint Meeting on European Software Engineering Conf. and Symp. Foundations of Software Engineering, pp. 739–743 (2018)

[17] Hailesilassie, T.: Rule extraction algorithm for deep neural networks: A review. CoRR **abs/1610.05267**, 555,555 (2016)

[18] Hu, Z., Ma, X., Liu, Z., Hovy, E.H., Xing, E.P.: Harnessing deep neural networks with logic rules. In: Proc. 54th Annu. Meeting of the Association for Computational Linguistics, vol. 1: Long Papers (2016)

[19] Huang, X., Kroening, D., Kwiatkowska, M., Ruan, W., Sun, Y., Thamo, E., Wu, M., Yi, X.: Safety and trustworthiness of deep neural networks: A survey. CoRR **abs/1812.08342** (2018)

[20] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial Examples Are Not Bugs, They Are Features. CoRR **abs/1905.02175** (2019)

[21] Johnson, C.W.: The increasing risks of risk assessment: On the rise of artificial intelligence and non-determinism in safety-critical systems. In: Evolution of System Safety: Proc. Safety-Critical Systems Symp. (2017)

[22] Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: Proc. 29th Int. Conf. Comput. Aided Verification, pp. 97–117 (2017)

[23] Kendall, A., Gal, Y.: What uncertainties do we need in Bayesian deep learning for computer vision? In: Advances in Neural Information Processing Systems 30, pp. 5580–5590 (2017)

[24] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., Sayres, R.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In: Proc. 35th Int. Conf. Machine Learning, vol. 80, pp. 2668–2677 (2018)

[25] Kim, J., Canny, J.F.: Interpretable learning for self-driving cars by visualizing causal attention. In: Proc. 2017 IEEE Int. Conf. Comput. Vision, pp. 2961–2969 (2017)

[26] Kim, J., Rohrbach, A., Darrell, T., Canny, J.F., Akata, Z.: Textual explanations for self-driving vehicles. In: Proc. 15th European Conf. Comput. Vision, Part II, vol. 11206, pp. 577–593 (2018)

[27] Kindermans, P.J., Schütt, K.T., Alber, M., Müller, K.R., Erhan, D., Kim, B., Dähne, S.: Learning how to explain neural networks: PatternNet and PatternAttribution. In: Proc. 6th Int. Conf. on Learning Representations (2018)

[28] Koopman, P., Fratrik, F.: How many operational design domains, objects, and events? In: Workshops of the 32nd AAAI Conf. Artificial Intelligence (2019)

[29] McAllister, R., Gal, Y., Kendall, A., van der Wilk, M., Shah, A., Cipolla, R., Weller, A.: Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. In: Proc. 26th Int. Joint Conf. Artificial Intelligence, pp. 4745–4753 (2017)

[30] Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill **2**(11), e7 (2017)

[31] Pei, K., Cao, Y., Yang, J., Jana, S.: Towards practical verification of machine learning: The case of computer vision systems. CoRR **abs/1712.01785** (2017)

[32] Petsiuk, V., Das, A., Saenko, K.: RISE: Randomized input sampling for explanation of black-box models. In: Proc. British Machine Vision Conf., p. 151 (2018)

[33] Rabold, J., Siebers, M., Schmid, U.: Explaining black-box classifiers with ILP – empowering LIME with Aleph to approximate non-linear decisions with relational rules. In: Proc. Int. Conf. Inductive Logic Programming, pp. 105–117 (2018)

[34] Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)

[35] Roychowdhury, S., Diligenti, M., Gori, M.: Image classification using deep learning and prior knowledge. In: Workshops of the 32nd AAAI Conf. Artificial Intelligence, vol. WS-18, pp. 336–343 (2018)

[36] Ruan, W., Huang, X., Kwiatkowska, M.: Reachability analysis of deep neural networks with provable guarantees. In: Proc. 27th Int. Joint Conf. Artificial Intelligence, pp. 2651–2659 (2018)

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

[38] Sensoy, M., Kaplan, L.M., Kandemir, M.: Evidential deep learning to quantify classification uncertainty. In: Advances in Neural Information Processing Systems 31, pp. 3183–3193 (2018)

[39] Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. CoRR **abs/1708.06374** (2017)

[40] Sun, Y., Wu, M., Ruan, W., Huang, X., Kwiatkowska, M., Kroening, D.: Concolic testing for deep neural networks. In: Proc. 33rd ACM/IEEE Int. Conf. Automated Software Engineering, pp. 109–119 (2018)

[41] Thrun, S.: Extracting rules from artificial neural networks with distributed representations. In: Advances in Neural Information Processing Systems 7, pp. 505–512 (1995)

[42] Wang, H.: ReNN: Rule-embedded neural networks. In: Proc. 24th Int. Conf. Pattern Recognition, pp. 824–829 (2018)

[43] Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: Proc. 27th USENIX Security Symp., pp. 1599–1614 (2018)

[44] Xiang, W., Musau, P., Wild, A.A., Lopez, D.M., Hamilton, N., Yang, X., Rosenfeld, J.A., Johnson, T.T.: Verification for machine learning, autonomy, and neural networks survey. CoRR **abs/1810.01989** (2018)

[45] Zhang, Q., Zhu, S.C.: Visual interpretability for deep learning: A survey. Front. IT EE **19**(1), 27–39 (2018)