https://doi.org/10.20378/irb-47274

Strategies for Safety Goal Decomposition for Neural Networks

Gesina Schwalbe
Continental Automotive GmbH
Artificial Intelligence and Robotics Labs
Regensburg, Germany
gesina.schwalbe@continental-corporation.com

Martin Schels
Continental Automotive GmbH
Artificial Intelligence and Robotics Labs
Regensburg, Germany
martin.schels@continental-corporation.com

ABSTRACT

Neural networks (NNs) have become a key technology for solving highly complex tasks, and require integration into future safety argumentations. New safety relevant aspects introduced by NN based algorithms are: representativity of test cases, robustness, inner representation and logic, and failure detection for NNs. In this paper, a general argumentation structure for safety cases respecting these four aspects is proposed together with possible sources of evidence.

KEYWORDS

neural network, safety argumentation, safety goal decomposition

ACM Reference Format:
Gesina Schwalbe and Martin Schels. 2019. Strategies for Safety Goal Decomposition for Neural Networks. In *Proceedings of ACM Computer Science in Cars Symposium (CSCS'19).* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Their outstanding performance in complex tasks such as in computer vision suggests that neural networks (NNs) will take a key role in enabling new technologies like autonomous driving. However, little experience is available on how to assure safety of NN based applications, and the existing safety standards give insufficient guideline on how to treat black-box algorithms which are not based on expert knowledge but learnt from data [26]. There are four new aspects of NNs that require consideration in a safety case:

Test case selection. Large and unstructured input spaces as NNs can be applied to do not admit easy reduction of test cases. A structured way of test case selection is necessary.

Robustness issues. NNs are prone to adversarial examples [2]. This means an imperceivably small change of a correctly treated input leads to drastically different, possibly unsafe behavior. Such a lack of smoothness around a sample invalidates it as a test case.

Black-box character. The black-box character of NNs can hide faults of the algorithm given by a wrong inner representation or logic. Such can be wrong causal relations learned from a bias in the training data.

Failure detection. NNs require technology specific failure indicators like uncertainty [17] which can be evaluated during runtime to enable effective risk reduction.

In the following, reusable ways to decompose safety goals related to NN algorithmic failures are developed respecting above aspects, and diverse ways to collect evidence are suggested. Safety goals and *strategies* are marked as indicated. The focus is solely on functional safety as in [26]. For safety of the intended functionality see [10]. We only discuss NN algorithms that are deterministic during inference, especially excluding algorithms learing during operation.

2 ARGUMENTATION STRUCTURE

Consider the traditional decomposition of a safety argumentation into process compliance and product compliance, which itself is split into an argument over each hazard [12, p. 14]. We suggest to argue the low probability of a hazard by *exercising through its possible sources.* Our interest lies in such sources that directly emanate from the algorithm. Due to the determinism assumption these only are systematic failures, i.e. wrong input to output mappings (errors). Hence, the base safety goal for our argumentation is that the risk of (considered) NN algorithmic errors is sufficiently low. *Measures that can be applied can be categorized* into error prevention on algorithm level and prevention and mitigation on system level.

2.1 Prevention of Errors on Algorithm Level

We identified three types of safety issues that can directly emanate from the algorithm: *performance, robustness, and logical issues.* These outline the pillars for decomposition.

2.1.1 *Validation of Inner Representation and Logic.*

To avoid logical issues, the NN should correctly use generalizable concepts and relations. This requires that both the inner representation used by the NN and the logic applied are sensible in the sense of good generalization capability. E.g. predicting a pedestrian next to a street light because of presence of limbs is more generalizable than just because of the relation to the street light.

We say a semantic concept is internally represented in the NN if the NN admits an intermediate output which is a prediction of this concept [8]. Essential concepts are ones considered necessary or helpful to solve the given task, e.g. because they are used in natural language descriptions [1]. A way to quantitatively assess or constructively modify the inner representation of NNs is concept embedding analysis [8], which finds or trains the neuron vector that best predicts a given visual semantic concept. Another way to enforce concepts is by adding functional neurons as in [22]. Qualitative assessments can be done by searching for typical input patterns activating given parts of the network. Methods for this are constructive using feature visualization [20], or analytic using attribution analysis e.g. heatmapping [13, 19].

A valuable inner representation alone does not ensure a correct and sensible internal logic. The approaches to ensure that sensible reasoning is applied are to quantitatively verify sensible domain knowledge based rules, or to qualitatively analyze whether the contained rules are sensible. Often, more domain knowledge is available than is presented to the NN during training, e.g. the physical extend of cars for trajectory planning or hierarchical relations like 'humans usually have heads'. Such domain knowledge rules can be expressed as first-order or temporal logic statements on the neurons of the NN given suitable concept embeddings. They then can either be formally verified using a solver [11]; topologically enforced [27]; or enforced during training by adapting either the loss [22], the training data, or by applying model or data repair [11]. Different ways to (partly) grasp the internal logic in an interpretable form are local approximations via e.g. qualitative attribution analysis to compare the NN attention with expected relevance of the input parts, additional output like textual explanations [18], or local rule extraction [21]; or global approximations by global rule [3] respectively automaton [28] extraction.

2.1.2 Verification of Robustness. There are several types of robustness in the context of NNs. They require the input and output spaces to be normed vector spaces. For y in a metric space M with metric d let $b_\alpha(y) = \{x \in M | d(x, y) < \alpha\}$. Then, the NN f is said to be locally δ-ϵ-robust around an input sample x against a perturbation $P \subset b_\delta(x)$, which is a parametrized subspace like a hyperplane, if

$$f(P) \subset b_\epsilon(f(x)) \quad \text{(compare [16]).}$$

This means there is an upper bound of 2ϵ to the change of the output of f between samples in P. f is globally δ-ϵ-robust against a perturbation $P \subset b_\delta(0)$ with respect to a subspace X of the input space if it is so locally against $P + x$ for all x in X. This is a weakened form of Lipschitz continuity, which again is a weakened form of smoothness. Given an ϵ, perturbations, and X, then robustness can be measured as the maximum δ allowing for local/global δ-ϵ-robustness against the perturbations. Since it is currently computationally impractible to assess robustness for the complete ball, we suggest to concentrate on and *exercise through task specific perturbations*, e.g. caused by changes in lighting condition, slight shifts of objects, dirt/rain drops on the camera, or deficiency of isolated pixels.

Measures to increase robustness against a given perturbation can be taken during model design, training, or verification: It is shown that certain network architectures, especially approaches for uncertainty modeling, yield more robust models [9, 24]. By definition, this is also the case for smoothing of the NN during training e.g. via regularization [15]. Retraining on adversarial counterexamples [6] has a similar, perturbation specific effect. Finally, one can disprove robustness properties using differential or fuzzy testing [14], or prove them using SMT-based [5] or MILP-based [7] solvers for formal verification.

2.1.3 Performance Assurance via Structured Testing. As for traditional software, the NN performance is measured and judged on a (representative) test set (different from the training set). It is crucial to *argue the representativity for all available aspects*, of which we identified coverage of previous experience to prevent regression, of the model behavior range, and of the input space respectively task in the sense of both completeness and realism.

For input space coverage, assume a formalization function F exists from the input space to a formal scenario description language consisting of countable discrete sets of concepts and relations thereon. Further a reconstruction function R is needed from formal scenario descriptions to input space, such that $R \circ F$ is idempotent, and $F \circ R$ does not exceed an expert chosen distance to the identity. Due to the discreteness, the amount of all scenarios (of interest for the hazard) is finite by combinatorics admitting a coverage metric, whilst the small distance to identity ensures realism. This would make it possible to systematically select or generate a realistic test set with desired completeness. An example of a formal description language for autonomous driving is developed in [4].

For model coverage, the test set should be augmented to fully cover the NN behavior range as measured by metrics such as neuron coverage [25]. And invaluable experience can be given as previously found error cases like adversarial examples or accidents from accident databases (e.g. [23]).

2.2 Failure Handling

To render remaining algorithm failures harmless, these must be handled correctly. *Aspects of failure handling* are: Prevention on system level, e.g. via input quality assurance, redundancy and safe-aware adaption of operating modes, detection, and mitigation on system level, e.g. via filtering of intermediate errors and an emergency backup system.

For the technology specific part of failure detection (logical, robustness) a couple of indicators are available: Plausibility checks (e.g. pedestrians usually do not fly) and attention monitoring, which compares the distance of output and attention positions, can indicate logical failures. Specialized heatmapping assists to reveal adversarial examples of certain types [13]. And low confidence respectively high uncertainty [9, 17, 24] can indicate a decision border respectively samples that are out of scope for the trained model.

3 CONCLUSION

Generic strategies were described on how to include key issues of NN algorithms into a safety argumentation by arguing over the safety relevant NN errors. Together with the diverse suggested sources of evidence, this promises to enable a complete safety case for NN based systems. Evaluation of our approach on a concrete example case is left for future steps.

REFERENCES

[1] Jacob Andreas, Dan Klein, and Sergey Levine, *Learning with latent language*, Proc. Conf. North Amer. Chapter of the Assoc. for Computational Linguistics: Human Language Technologies, vol. 1, 2018, pp. 2166–2179.

[2] Felix Assion, Peter Schlicht, Florens Greßner, Wiebke Günther, Fabian Hüger, Nico Schmidt, and Umair Rasheed, *The Attack Generator: A Systematic Approach Towards Constructing Adversarial Attacks*, Proc. 2019 IEEE Conf. Comput. Vision and Pattern Recognition Workshops, 2019.

[3] M. G. Augasta and T. Kathirvalavakumar, *Rule extraction from neural networks — A comparative study*, Proc. 2012 Int. Conf. Pattern Recognition, Informatics and Medical Engineering, 2012, pp. 404–408.

[4] G. Bagschik, T. Menzel, and M. Maurer, *Ontology based scene creation for the development of automated vehicles*, Proc. 2018 IEEE Intelligent Vehicles Symp., 2018, pp. 1813–1820.

[5] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda, *A unified view of piecewise linear neural network verification*, Advances in Neural Information Processing Systems 31, 2018, pp. 4790–4799.

[6] Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Kurt Keutzer, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia, *Counterexample-guided data augmentation*, Proc. 27th Int. Joint Conf. Artificial Intelligence, 2018, pp. 2071–2078.

[7] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari, *Output range analysis for deep feedforward neural networks*, Proc. 10th Int. Symp. NASA Formal Methods, vol. 10811, 2018, pp. 121–138.

[8] Ruth Fong and Andrea Vedaldi, *Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks*, Proc. 2018 IEEE Conf. Comput. Vision and Pattern Recognition, 2018, pp. 8730–8738.

[9] Jochen Gast and Stefan Roth, *Lightweight probabilistic deep networks*, Proc. 2018 IEEE Conf. Comput. Vision and Pattern Recognition, 2018, pp. 3369–3378.

[10] Lydia Gauerhof, Peter Munk, and Simon Burton, *Structuring validation targets of a machine learning function applied to automated driving*, Computer Safety, Reliability, and Security, 2018, pp. 45–58.

[11] Shalini Ghosh, Patrick Lincoln, Ashish Tiwari, and Xiaojin Zhu, *Trusted machine learning: Model repair and data repair for probabilistic models*, Workshops 31st AAAI Conf. Artificial Intelligence, vol. WS-17, 2017.

[12] SCSC Assurance Case Working Group, *Goal structuring notation community standard*, jan 2018 ed., 2018.

[13] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi, *A survey of methods for explaining black box models*, ACM Comput Surv **51** (2018), no. 5, 93:1–93:42.

[14] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun, *DLFuzz: Differential fuzzing testing of deep learning systems*, Proc. ACM Joint Meeting on European Software Engineering Conf. and Symp. Foundations of Software Engineering, 2018, pp. 739–743.

[15] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu, *Safety verification of deep neural networks*, Proc. 29th Int. Conf. Comput. Aided Verification, Part I, vol. 10426, 2017, pp. 3–29.

[16] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer, *Reluplex: An efficient SMT solver for verifying deep neural networks*, Proc. 29th Int. Conf. Comput. Aided Verification, 2017, pp. 97–117.

[17] Alex Kendall and Yarin Gal, *What uncertainties do we need in Bayesian deep learning for computer vision?*, Advances in Neural Information Processing Systems 30, 2017, pp. 5580–5590.

[18] Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John F. Canny, and Zeynep Akata, *Textual explanations for self-driving vehicles*, Proc. 15th European Conf. Comput. Vision, Part II, vol. 11206, 2018, pp. 577–593.

[19] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne, *Learning how to explain neural networks: PatternNet and PatternAttribution*, Proc. 6th Int. Conf. on Learning Representations, 2018.

[20] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert, *Feature visualization*, Distill **2** (2017), no. 11, e7.

[21] Johannes Rabold, Michael Siebers, and Ute Schmid, *Explaining black-box classifiers with ILP – empowering LIME with Aleph to approximate non-linear decisions with relational rules*, Proc. Int. Conf. Inductive Logic Programming, 2018, pp. 105–117.

[22] Soumali Roychowdhury, Michelangelo Diligenti, and Marco Gori, *Image classification using deep learning and prior knowledge*, Workshops of the 32nd AAAI Conf. Artificial Intelligence, vol. WS-18, 2018, pp. 336–343.

[23] A. Schubert, H. Liers, and M. Petzold, *The GIDAS pre-crash-matrix 2016. Innovations for standardized pre-crash-scenarios on the basis of the VUFO simulation model VAST*, Reports on the ESAR-Conference 2016, vol. 117, 2017, pp. https://www.gidas.org/en/gidas–publikationen/.

[24] Murat Sensoy, Lance M. Kaplan, and Melih Kandemir, *Evidential deep learning to quantify classification uncertainty*, Advances in Neural Information Processing Systems 31, 2018, pp. 3183–3193.

[25] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening, *Concolic testing for deep neural networks*, Proc. 33rd ACM/IEEE Int. Conf. Automated Software Engineering, 2018, pp. 109–119.

[26] Stefan Voget, Alexander Rudolph, and Jürgen Mottok, *A consistent safety case argumentation for artificial intelligence in safety related automotive systems*, Proc. 9th European Congress on Embedded Real Time Systems, 2018.

[27] Hu Wang, *ReNN: Rule-embedded neural networks*, Proc. 24th Int. Conf. Pattern Recognition, 2018, pp. 824–829.

[28] Qinglong Wang, Kaixuan Zhang, Alexander G. Ororbia II, Xinyu Xing, Xue Liu, and C. Lee Giles, *An empirical evaluation of rule extraction from recurrent neural networks*, Neural Comput. **30** (2018), no. 9, 2568–2591.