Concept Enforcement and Modularization as Methods for the ISO 26262 Safety Argumentation of Neural Networks

Gesina Schwalbe and Martin Schels

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

Abstract. Neural networks (NN) are prone to systematic faults which are hard to detect using the methods recommended by the ISO 26262 automotive functional safety standard. In this paper we propose a unified approach to two methods for NN safety argumentation: Assignment of human interpretable concepts to the internal representation of NNs to enable modularization and formal verification. Feasibility of the required concept embedding analysis is demonstrated in a minimal example and important aspects for generalization are investigated. The contribution of the methods is derived from a proposed generic argumentation structure for a NN model safety case.

Keywords: concept enforcement, machine learning, neural networks, functional safety, ISO 26262, goal structuring notation, explainable AI

1 Introduction

The complexity and black-box nature of NNs makes the suggested set of methods from the ISO 26262 automotive functional safety standard insufficient for plausible safety assurance. This is e.g. the case for autonomous driving perception functions, or in general computer vision applications solved with convolutional neural networks (CNN) as a safety critical function, which is the scope of this paper.

We propose a template for the part of the safety case concerning model assessment in order to identify needed methods. One is *modularization* of NNs, i.e. simplification or reduction, e.g. pruning or splitting into environment abstraction and trajectory planning instead of an end-to-end approach. For a useful split it is necessary that the interface of the sub-modules as well as their intended functionality is clear. Instead of using a fixed modular topology as in [25], we propose to utilize inherent modules of a trained NN by identifying interpretable intermediate outputs as splitting points.

One source of evidence for a safety case is formal verification, which requires a formal language on the available input and (intermediate) output of the algorithm in order to define the rules to be verified. A formal language consists

of words (human interpretable concepts) and valid relations thereon, which are both seldom available for neural networks. Therefore, we suggest as a second needed method the dynamic *enforcement* of a preselected set of interpretable intermediate outputs. These can then serve as vocabulary for a formal description language and should be build up from semantic *concepts*, i.e. abstract classes of real world objects (e.g. body parts like "foot") or attributes (e.g. textures like "hairy"), which admit real world *relations* (e.g. spatial or hierarchical) between them.

The Contributions of this paper are:

- A generic model assessment part for the safety argument of NN based applications is provided.
- A theory on embeddings of visual semantic concepts into NNs is developed.
- The safety argumentation template is enriched by a unified approach for concept enforcement and modularization which based on the theory of concept embeddings. For each a workflow is suggested.
- An approach for concept embedding analysis is investigated.

2 A Safety Argumentation Structure for Neural Networks

A template for arguing the safety of the intended functionality of NN based systems can be found in [12]. Development of functional safety argumentation started with very basic considerations in [18], revised and more detailed in [24]. Contrary to the latter and inspired by the template suggested in [13, p. 14], we propose to more concretely split between product based (verification and validation) and process based argumentation. In the following a template, for product based safety argumentation of NN based algorithms is suggested with focus on the product argumentation regarding NN specific properties. As in the preliminary work, goal structuring notation [13] is used, which is regarded common practice for safety argumentation [9].

Figure 1 proposes a structure for the NN specific functional safety parts of a safety argumentation. The idea of decomposition is to split into product (**S1**) and process (**G2**) argumentation. Within the process argumentation (**S2**) we locate the need for a modularization strategy for overly complex models (**Sn1**): Due to the black-box character and complexity of NNs, the effort for the corresponding safety argumentation increases exponentially with the size of the network [14].

A NN product argumentation requires special care compared to traditional software models. Reasons are that the intended functionality might be little understood, that a high probability of remaining systematic faults requires proper safety mechanisms (**G3**), and that the absence of safety relevant systematic faults introduced by NN specific problems (**G4**) needs to be proven, but such are not considered in the ISO 26262 standard so far. Figure 1 concentrates on an argumentation strategy which can provide reasonable confidence for the last goal.

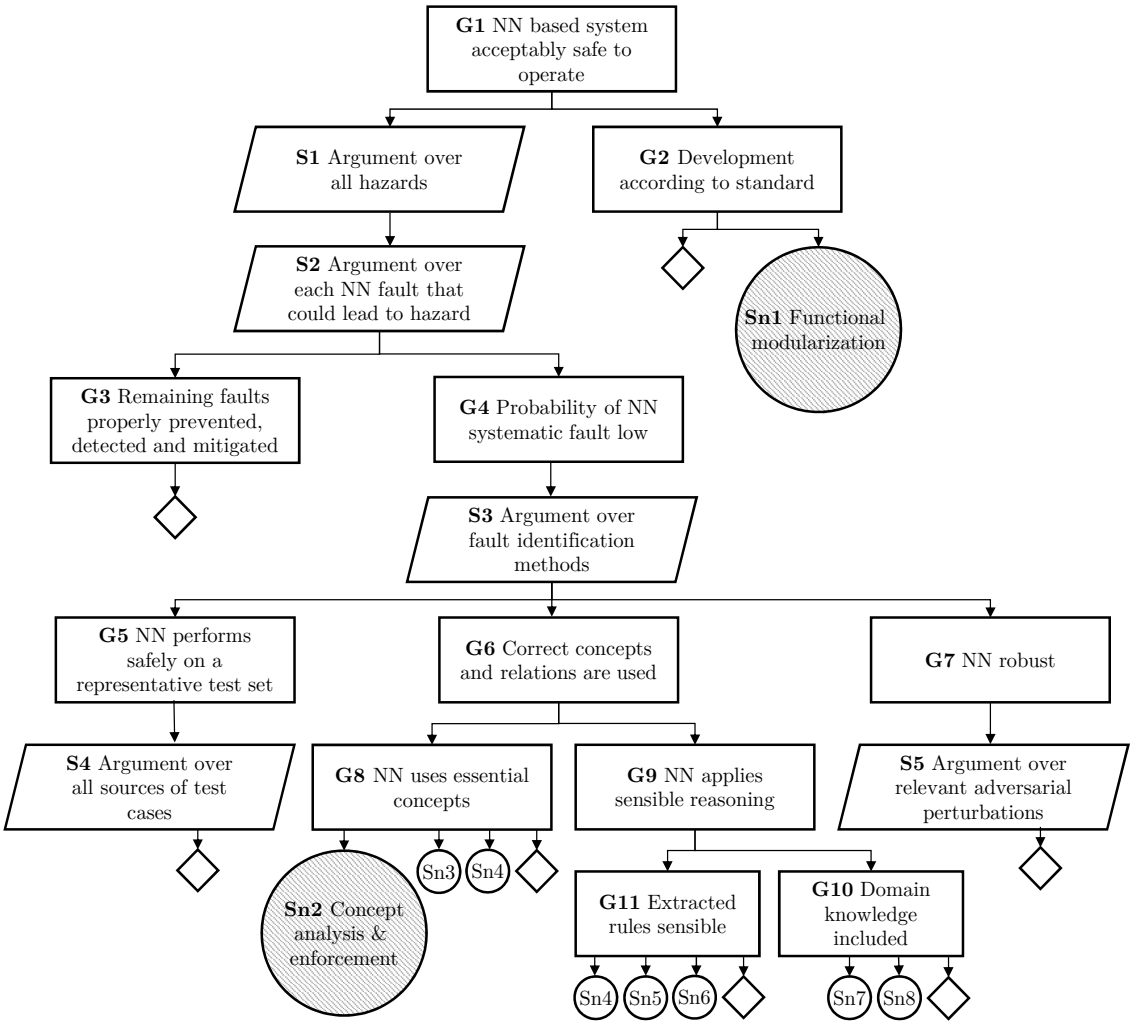

Fig. 1: Template excerpt from a safety case structure for a NN algorithm, focusing on product argumentation regarding NN model specifics. Sub-trees where we see suggest further methods (modularization and concept enforcement) are colored. Parts: safety (sub-)goals (**G*X***), argumentation strategies (**S*X***), methods to provide evidence (**Sn*X***), and arrows depict dependencies/break downs. Skipped parts are marked with diamonds.

We categorize NNs specific failure modes into three categories: Robustness against small input changes, wrong internal representation or logic of the black-box which we are dealing with in our method proposal, and bad generalization performance due to training data representativity. We suggest that faults of NNs (**S3**) leading to one of those failure modes can be found by one of the following verification and validation approaches: One is thorough testing (**G5**) on systematically acquired test cases (**S4**). Another aspect is robustness assurance (**G7**) for all relevant input perturbations (**S5**). Lastly, due to the complexity of the considered input space, we claim that evidence for a sensible internal structure of the algorithm (**G6**) is required. Sensible here means beneficial or necessary for the intended functionality according to expert judgment. For example classifying speed signs requires the ability to detect and distinguish digits.

The internal structure includes the internal representation (**G8**) and the internal logic (**G9**) applied to it. The latter can either be directly assessed (**G11**) via local (**Sn5**, e.g. [20], [16]) or global (**Sn6**, see [4]) rule extraction, or can be verified by comparison with given rules (**G10**), which can be done by formal verification (**Sn8**, e.g. [7]), or by enforcement of rules (**Sn7**, e.g. via loss [21]). Similar to rule extraction for logic assessment, qualitative methods for inner representation assessment are available. Such are analysis of attention (**Sn4**, e.g. [17]), or analysis of inherent intermediate features (**Sn3**, e.g. [19]). However, we identified a lack of quantitative methods for the analysis and improvement of domain knowledge usage in the inner representation (**Sn2**). Method suggestions for analysis were given by [10, 15]. Our quantitative proposal of concept enforcement builds upon these and fills a gap here.

3 Unified Approach for Concept Enforcement and Modularization

As will be detailed in the following section, semantic concepts are naturally embedded into the internal representations of NNs. We suggest to utilize this property in two directions: Given desirable concepts, assess and enforce their usage. And, knowing which ones are represented internally, use these as splitting points for a modularization strategy.

3.1 Background Theory: Concept Embeddings

Consider the following definitions generalized from [15]. The intermediate output of a NN layer (or, in general, of a set of independent neurons) is a vector within the space of neuron outputs, here called the *space of abstract features* of the layer. Consider a layer, an abstract feature vector w, the projection p_w to the one dimensional sub-vector space spanned by x, and a concept c. The vector w is said to be a *concept embedding* of c if the intermediate output of the NN obtained by concatenation of the layer with p_w has a high correlation with the existence of c (in a certain spatial location). Simply speaking, w is the normal vector of the hyperplane separating the layer output into c or not-c.

It was shown that there often are concepts embedded by the unit vectors (i.e. by single neurons) [6], and that for many task related concepts embeddings can be identified as investigated in [10, 15]. They furthermore found that similar concepts are embedded by close vectors, and vector operations yield meaningful relations on the concepts, and that concept vectors can mostly be chosen sparse [10], i.e. simple basis vector combinations. Sparsity of concepts possibly gives a measure of encoding complexity and of alignment of the layer basis to the human concept base. So, a NN naturally internally represents and uses concepts, and these can be extracted as additional output. Note that previous literature was restricted to feature spaces consisting of neuron outputs of exactly one layer, not general collections of independent neurons, but the definitions and possibly the properties generalize to such.

The above approach to concept embedding analysis gives rise to valuable metrics: The correlation value of a concept embedding serves as quality metric for the embedding. The quality of the best concept embedding for a concept shows *how well the concept is embedded* into the network. Given an input, the additional output by projection to the concept embedding shows *how well the network detected the concept for the given input instance*. And directed derivative along the vector shows *the attribution of the concept towards the decision* [15].

3.2 Concept Enforcement

The inclusion of task related natural language concepts and outputs has been shown to improve the performance of NNs: Loss based examples are fuzzy logic rules on hierarchical concepts [21] and multi-task learning including sub-tasks [11]. Other ways of rule enforcement are e.g. inclusion it into the data [8], topologically [25], or in the case of reinforcement learning safe learning [2]. A possible reason for the positive impact is that natural language in general admits highly efficient abstraction levels for real world tasks [3].

Explainable intermediate output as given by concept embeddings enables one to formulate and automatically verify requirements on the NN. The above suggests that enforcement of output of preselected concepts can be realized without negative impact on the performance, thus qualifies as post-training fine-tuning method. We propose the following workflow:

1. *Identification* of task relevant concepts and relations: We propose for a start two criteria to identify relevant concepts and relations: They are used or needed for synthetic data generators, i.e. are used for the underlying ontology as in [5]. Or, they are used by NNs in similar tasks. Concepts used by NNs can be identified using concept embedding analysis on predefined concepts, or methods like explanatory graph extraction [27] that need manual assignment of natural language descriptions to the found concepts. Used relations can be found by inspecting the embedding vectors or rule extraction methods like [20] which uses inductive logic programming.
2. *Definition of a formal description language* out of these concepts and relations: The previous guidance on vocabulary selection should ensure a domain relevance.

3. *Formulation of rules* using this formal language: This is highly use-case specific and requires domain knowledge. General aspects to derive rules from can be plausibility checks, e.g. derived from physical laws ("pedestrians usually don't fly") or hierarchical relations ("children and adults are pedestrians", "a head usually belongs to a pedestrian"). Or they can be derived from safety bounds, e.g. performance guarantees ("pedestrian detection performance independent of clothing color") or safe state guarantees ("trajectories keep safe distance from obstacles").
4. *Verification* of these rules (automatically) using available solvers, a nice overview of which can be found e.g. in [7]: Current approaches differ in performance, restrictions to the network activation functions, and the rules that can be checked; notably upper/lower bound rules verifiable via boundary approximation [26] versus general linear constraints that are solved by satisfiability modulo theory based approaches [7].
5. *Enforcement* of rules if necessary: Of the methods suggested above retraining with counterexamples and a modified loss are most promising for the chosen example use-case discussed later.

3.3 Modularization

Splitting at interpretable intermediate outputs results in several smaller and, hence, less complex sub-networks respectively functional modules, which can be iteratively optimized and verified. This is eased by reducing the number of neurons per sub-network and intersection of such using pruning of the neuron connections before splitting, i.e. deleting ones with low weight. We suggested the following recursive workflow to achieve higher modularization:

1. *Identification of learned concepts* using neuron level analysis, e.g. [6, 10].
2. *Radical pruning* of connections, e.g. weight decay and thresholding [1, 22].
3. *Identification* of the abstraction and interpretation networks using topological introspection (if possible automatically).
4. *Splitting* of the NN into the smaller NNs with the identified interfaces.
5. *Evaluation* and (partly) *retraining* of the pruned and split NN if necessary.
6. *Simplified safety assessment* on the smaller parts.

4 Experiment on Concept Embedding Analysis

The idea on how to obtain a concept embedding vector is to train it and its threshold as parameters for a linear predictor on the model intermediate output. In [15] they use a support vector machine as predictor, which improves the uniqueness of results. The approach in [10] uses a convolutional layer for the prediction introducing translation invariance, easy sparsity analysis, and the possibility of concept segmentation. We base on this setup, since it can directly be used for a multi-task training problem to enforce the desired concept. We tested whether the concept embedding analysis suggested in [10] generalizes to a minimal example and investigated several proposals for improvement.

General Setup. The setup is a simple traffic sign classifier on the subset of 15 classes of the German Traffic Sign Recognition dataset [23] with all images scaled to 48×48 px. The visual semantic concepts to analyze were the digits occurring in the five different speed limit sign classes. A concept training respectively testing set are all images containing the digit together with a random sampling of others at a ratio of 7:3. Due to the sign uniformity and translation invariance, the concepts were statically labeled. The pretrained classifier was realized at accuracy of 98.2 % with a feed-forward NN with four convolutional blocks, each ReLU-activated 3×3-convolution and 2×2 max pooling, then two dense blocks, and a final dense sigmoid layer. Each block is succeeded by a dropout layer. The intermediate output wherein to find a concept embedding was chosen to be a window of 3×4 pixels (width $\times$ height) in the activation map of the third convolutional layer. This differs from [10] where only one pixel was considered at a time because it proved to be necessary that the receptive field of the considered window (when up-scaling the window pixels to the original image) covers the complete digit, or, more generally, uniquely identifying parts of the concept. For example a "3" and "5" cannot be distinguished by their lower half. Similarly, the training objective we found to generalize best differs fundamentally from [10]: For a position, we want to predict whether it is the center of a window containing the desired concept, other than directly creating a pixel-wise mask. For translation invariance, the window setup was realized by a 3×4-convolution with one output filter (the concept output) and sigmoid activation on the output of the layer. The concept embedding vector to train is the weight vector of the convolution kernel. For evaluation of the embedding respectively the associated classifier we used smooth set intersection over union (siou) as in [6, 10] which is calculated as the sum of intersections of the predicted mask $M(x)$ with the ground truth mask $M_{\text{gt}}(x)$ for all samples x in the test set X

$$\text{siou}(X) = \frac{\left(\sum_{x \in X} M_{\text{gt}}(x) M(x)\right) + 1}{\left(\sum_{x \in X} M_{\text{gt}}(x) + M(x) - M_{\text{gt}}(x) M(x)\right) + 1} \,. \tag{1}$$

Proposals for Improvement. As losses we compared the standard losses binary cross-entropy (bc) and mean squared error (mse) to our new suggestions of negative smooth set IoU from (1) with $M(x)$ non-binary, and a variant derived from the source code of [10] (si), which optimizes for large intersections each of foreground and of background pixels:

$$\text{si}(B) = -\tfrac{1}{\#B} \sum_{x \in B} (1-b) M_{\text{gt}}(x) M(x) + b \left(1 - M_{\text{gt}}(x)\right) \left(1 - M(x)\right)$$

where B is a batch and b is the mean number of foreground pixels in the dataset as a weighting factor. This weighting factor (w) as well as smoothing the objective by predicting the intersection over union (IoU) value of a window (iou) were tested for performance benefits. Lastly, we investigated whether convergence could be improved by pretraining the weight vector in a digit classifier setup: A balanced set of windows with and without the digit is presented to the network. Due to the convolution size, this yields one classification output pixel per image.

Loss	pretrained	
	yes	no
si	0.313	0.016
si-w	0.308	0.138
si-iou	0.305	0.011
si-w-iou	0.386	**0.200**
siou	0.264	**0.325**
siou	0.047[a]	0.093[a]
bc	**0.473**	0.094
bc-w	–	–
bc-iou	**0.421**	0.050
bc-w-iou	–	**0.198**
mse	**0.423**	0.025
mse-w	0.223	0.099

[a] pixels binarized, not bloated

Fig. 2: Top to bottom: Originals, ground truth IoU masks, exemplary outputs of the bc model with weight pretraining.

Table 1: Mean smooth set IoU results by setup (⟨loss⟩-⟨modifiers⟩) for concept "3"; gaps are numerically instable.

Results. Table 1 collects the mean set IoU of 10 runs for each setup (variance each below 0.004). For evaluation, the predicted IoU values were turned into an estimated concept area by up-scaling, bloating and then adding the IoU pixel values. For weight pretraining, the mean init model accuracy was 91%. The weighting w was originally suggested to be applied to bc instead of si loss in [10], which we found to be numerically instable. In general, w and iou were shown to only optimize si loss. Pretraining the weights on a concept training set with equal class distribution generally benefits the performance, supporting the suggestions from [10, 15]. Interestingly, pretraining the weights pushes bc and mse from no convergence to best results, suggesting that the losses themselves are weaker aligned with the optimization objective. We claim that this problem is caused by giving little account to spatial distance to the ground truth center, also in the iou setting. This well fits the observation that the direct application of siou loss, where the problem is mitigated, yields best results without any pretraining. Interestingly, siou will focus on finding pixels strictly inside the concept area, which can be seen from the comparison of IoU measurement directly on the output and after bloating the pixels. Finally, we found that all converging methods produce sparse concept vectors, since more than 50 % of the weight entries can be zeroed without inflicting but even increasing the performance. Therefore, we suggest that sparsity could be improved or even enforced based on above methods.

5 Conclusion and Outlook

The two suggested methods and workflows based on concept embedding analysis promise to substantially support a safety argument (Figure 1). The experiment results give guidance on generalizing an analysis approach from [10]. Future work will focus on applying these to the more complex safety relevant use-case of pedestrian detection. The final goal is to finish the goal structuring notation tree for the safety argumentation as a template for safety critical NN applications.

Bibliography

[1] Abbasi-Asl, R., Yu, B.: Interpreting convolutional neural networks through compression. CoRR **abs/1711.02329** (2017)

[2] 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)

[3] Andreas, J., Klein, D., Levine, S.: Learning with latent language. In: Proc. Conf. North Amer. Chapter of the Assoc. for Computational Linguistics: Human Language Technologies, vol. 1, pp. 2166–2179 (2018)

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

[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] Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proc. 2017 IEEE Conf. Comput. Vision and Pattern Recognition, pp. 3319–3327 (2017)

[7] 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)

[8] 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)

[9] Duffau, C., Polacsek, T., Blay-Fornarino, M.: Support of justification elicitation: Two industrial reports. In: Advanced Information Systems Engineering, vol. 10816, pp. 71–86 (2018)

[10] 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)

[11] 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)

[12] Gauerhof, L., Munk, P., Burton, S.: Structuring validation targets of a machine learning function applied to automated driving. In: Computer Safety, Reliability, and Security, pp. 45–58 (2018)

[13] Group, S.A.C.W.: Goal Structuring Notation Community Standard, jan 2018 edn. (2018)

[14] 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)

[15] 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)
[16] 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)
[17] 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)
[18] Kurd, Z., Kelly, T.: Establishing Safety Criteria for Artificial Neural Networks. In: Knowledge-Based Intelligent Information and Engineering Systems, pp. 163–169 (2003)
[19] Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill **2**(11), e7 (2017)
[20] 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)
[21] 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)
[22] Setiono, R., Liu, H.: Symbolic representation of neural networks. IEEE Comput. **29**, 71–77 (1996)
[23] Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German Traffic Sign Recognition benchmark: A multi-class classification competition. In: Proc. 2011 Int. Joint Conf. Neural Networks, pp. 1453–1460 (2011)
[24] Voget, S., Rudolph, A., Mottok, J.: A consistent safety case argumentation for artificial intelligence in safety related automotive systems. In: Proc. 9th European Congress on Embedded Real Time Systems (2018)
[25] Wang, H.: ReNN: Rule-embedded neural networks. In: Proc. 24th Int. Conf. Pattern Recognition, pp. 824–829 (2018)
[26] 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)
[27] Zhang, Q., Cao, R., Shi, F., Wu, Y.N., Zhu, S.C.: Interpreting CNN knowledge via an explanatory graph. In: Proc. 32nd AAAI Conf. Artificial Intelligence, pp. 4454–4463 (2018)