Monitoring Safety of Autonomous Vehicles with Crash Prediction Network

Saasha Nair, Sina Shafaei, Stefan Kugele, Mohd Hafeez Osman and **Alois Knoll**
Technical University Munich
Munich, Germany
saasha.nair@tum.de, sina.shafaei@tum.de, stefan.kugele@tum.de, hafeez.osman@tum.de, knoll@in.tum.de

Abstract

Automation needs safety inbuilt in the system such that it behaves at least as well as a diligent human in unforeseen circumstances, if not better. It is therefore necessary that the machine learns to behave intuitively by predicting future occurrences and take actions accordingly. Machine learning techniques, therefore, have to focus on safety issues. Human development and the consequential environmental changes will only push safety requirements higher demanding artificial intelligence to fill in the voids so generated. The purpose of this paper is to study the artificial intelligence perspective on safety challenges and concerns of such systems through an extensive literature review and propose a futuristic and easily adaptable system using deep learning technique. The paper would focus primarily on safety aspects of autonomous vehicles using Bayesian Deep learning method.

Introduction

Current trends in the automotive industry are introducing new, increasingly complex software functions into vehicles (Broy 2006). The ever-growing availability of computing resources, memory, and newest technologies allows for new levels of automated and intelligent systems. Driving at a high level of driving automation (i. e., level 3 to 5) according to SAE J3016 (Committee and others 2014) is just one example that has been discussed recently and is no longer just a future vision. Vehicles driving at levels 3 to 5 will be, hereafter, referred to as Autonomous Vehicles (AV) and the corresponding task as Autonomous Driving (AD).

Success stories in deep learning have made AVs more or less a reality, however, commercializing such vehicles have not yet fructified. Recent accidents, especially those involving cars driving at as low as SAE Level 2 show, that there are challenges engineers are still faced with, and the major impediment that stands in the way of large-scale adoption of AVs, is its associated safety concerns (Kalra and Paddock 2016; Fagnant and Kockelman 2015; McAllister et al. 2017).

Although there is no concrete solution in addressing the safety concern, several researchers have outlined the safety challenges and proposed recommendations to consider. Salay et al. (Salay, Queiroz, and Czarnecki 2017) analyzed the impact that the use of ML-based software has on various parts of ISO 26262 especially with respect to hazard analysis and risk assessment (HARA). Within the scope of highly automated driving (i. e., level 4), Burton et al. (Burton, Gauerhof, and Heinzemann 2017) explored the assurance case approaches that can be applied to the problem of arguing the safety of machine learning. From the ISO 26262 V-model perspective, Koopman and Wagner (Koopman and Wagner 2016) identified several testing challenges for autonomous vehicles. Monkhouse et al. (Monkhouse et al. 2017) reported several safety concerns to ensure the safety of highly automated driving from the functional safety engineers' perspective. This paper explores the challenges in developing and monitoring AI-based component for an end-to-end deep learning AV. The presented approach can minimize the apparent risk when dealing with machine learning based components of Autonomous Driving. However, a more fine-grain safety assessment such as safety requirements and risk assessment remain for future work. Basically, this research endeavors to answer the following research questions (RQ):

RQ1 What are the challenges involved in ensuring safety of highly critical systems when augmented with machine learning based components?

RQ2 What are the existing approaches used to ensure safety of learning systems?

RQ3 What are the shortcomings of the existing approaches and how can they be overcome?

Traditional Safety Techniques and Neural Networks

The main challenges (**RQ1**) associated with applying traditional safety assurance methodologies to NNs as it was explained in (Cheng et al. 2018) are as follows:

(i) ***Implicit Specification*** – Traditional Verification and Validation (V&V) methods (as suggested in ISO 26262 V model) lay great importance on ensuring that the functional requirements specified at the *design-time* of the system are met. However, NN-based systems depend solely on the training data for inferring the specifications of the model and do not depend on any explicit list of requirements, which can be problematic while applying traditional V&V methods. (ii) ***Black-Box Structure*** – While writing the code for a NN, one specifies the details about the layers and the activation functions, but, unlike traditional software, the control flow is not explicitly coded, leading to NNs being referred to

as black-box structures. Traditional white-box testing techniques such as code coverage and decision coverage cannot be directly applied to NNs, thus, there is a need to construct paradigms for adaptive software systems.

Related Work

We distinguish between the existing approaches by categorizing them into two groups: (i) 'Training phase', i. e.approaches that are solely used during the development and training phase of the neural network, and (ii) 'Operational phase', i. e.those that are used in the run-time environment of the neural network to ensure proper functioning (**RQ2**).

Training Phase

The existing approaches that fall under this category are:
(i) ***Train/Validation/Test split*** – This method is used to ensure that the developed adaptive system works satisfactorily for a given set of inputs. The method involves splitting the available data, to obtain three sets, such that the largest of the sets is used solely for training, and of the remaining two sets, one is used for fine tuning the hyperparameters of the NN, and the second one is used to test the working neural network to study how well it reacts to previously unseen data points. Though this method helps verify the working of the NN, it is not extensive enough to be considered a guarantee for safety (Taylor, Darrah, and Moats 2003) in high-criticality systems.
(ii) ***Automated test data generation*** – Lack of trust in the train-validation-test split method roots from the fact that one is left with very few data samples to test against, wherein, the chances are that cases of high interest might even get missed in the testing phase. A way to overcome this problem is to use test data generation tools to generate synthetic data points which, can be used for testing the trained neural networks. Tools such as Automated Test Trajectory Generation (ATTG) (Taylor 2006) and the more recent approach of generating scenes that an AV might encounter using ontologies (Bagschik, Menzel, and Maurer 2017) fall under this category. This approach can help the V&V procedure for NNs by unveiling missing knowledge in fixed NNs and increasing confidence in the working of adaptive NNs (Taylor, Darrah, and Moats 2003).
(iii) ***Formal Methods*** – Formal verification (Ray 2010) refers to the use of mathematical specifications to model and analyse a system. Though these methods work well with traditional software, they have not shown much success in the area of adaptive software systems. This is due to challenges (Seshia, Sadigh, and Sastry 2016) in modeling the non-deterministic nature of the environment, difficulty in establishing a formal specification to encode the desired and undesired behavior of the system, and the need to account for adaptive behavior of the system. Formal verification techniques for NNs deal instead with proving convergence and stability (Fuller, Yerramalla, and Cukic 2006) of the system, using methods such as Lyapunov analysis (Yerramalla et al. 2003).
(iv) ***Rule extraction*** – Rules (Darrah and Taylor 2006) are viewed as a descriptive representation of the inner workings of a neural network. Rule extraction algorithms, such as KT (Fu 1994), Validity Interval Analysis (VIA) (Thrun 1995), DeepRed (Zilke, Mencía, and Janssen 2016), can be used to model the knowledge that a neural network has acquired during the training phase. These rules can be expressed as easy to understand 'if-then' statements, that can either be manually verified owing to the human-readable format or can be automated with a model checker. This method can be helpful to establish trust in the system, as it augments the explainability of the system (Gasser and Almeida 2017). It also aids requirements traceability, as one can verify if the rules depict functional requirements specified for the system. They can also help to examine the various functional modes of the system and ensure that a safe operation mode is induced by certain inputs, while respecting the expected safety limits. Though this method brings in enormous advantages, it is more applicable for *offline learning* systems, wherein the V&V practitioner can extract rules from the network after training is complete.

Operational Phase

The solutions that fall under this category can be more accurately referred to as 'Online monitoring techniques', that involve the use of one or more monitors working as an oracle to ensure continued proper functioning of the neural network over time (Cukic et al. 2006). The goal here is to ensure that the adaptation dynamics does not cause the network to diverge, thereby triggering unpredictable behavior.

Data Sniffing (Liu, Menzies, and Cukic 2002) is an example based on the foregoing technique, which studies the data entering and exiting a neural network. If a certain input could pose negative results, then the monitors generate an alert and could even possibly flag down the data, thereby not allowing it to enter the system. This method is extremely useful in cases where outliers could degrade the functioning of the system.

Proposed Approach

Majority of the contemporary approaches, as evident from "Related Work" section, relate to testing a developed model before it is deployed in the operational environment. ML-based components, however, suffer from problems like; operational data/platform being different from what the model was trained on, uncertainty about the new inferences gained from operational data, and even wear-and-tear of hardware/software. This leaves ML-based components vulnerable to errors. Thus, it is necessary to focus on monitoring-based approaches, which are starting to gain recent interest (Fridman, Jenik, and Reimer 2017), to help alleviate the safety concerns associated with such systems.

To elaborate on specifics of the proposed solution, an end-to-end deep learning model for lane change maneuvers has been chosen. Such a model uses a deep neural net that takes input data from sensors that represent the environment around the ego vehicle, and generate one of three actions allowing the ego vehicle to continue driving in the current lane or to switch to the left or right lane depending on the presence of obstacles.

This proposed solution, referred to as 'Crash Prediction Network', involves a neural network model, tasked with de-

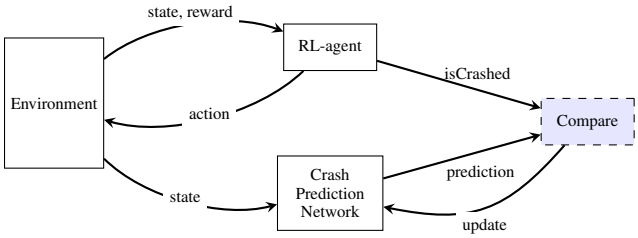

Figure 1: Training of Crash Prediction Network

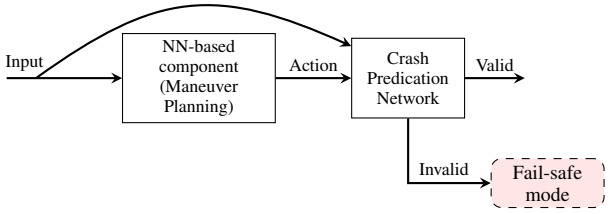

Figure 2: Operation of Crash Prediction Network

termining the likelihood and severity of a crash at any given time step (**RQ3**). The model takes into consideration multiple features such as output of the perception module of the vehicle, planned trajectory/action of the ego vehicle, predicted (or intended, if available via V2V communication) trajectory of the obstacles, and possibly also information such as number and severity of previous crashes that the ego vehicle and obstacles were involved in. Specifics of the system can be understood by distinguishing between the training and operational (after deployment) phases of the model.

The training phase (as shown in Fig. 1) relies on the model receiving the required input values for the previously described feature set, and also knowing whether a crash occurred or not. Thus, the model requires an architecture that involves a Reinforcement Learning environment, that would allow the model to know the outcome at every time step for a given set of feature values. This would also allow the vehicle to crash often, as is characteristic of RL-agents, especially at the start of training. We, therefore, propose to train the model by allowing it to spar with an RL-agent such that the ego vehicle closely imitates a real-world vehicle that can perform tasks similar to the lane change maneuver use-case described above. At each time step, the RL-agent and the Crash Prediction Network will have access to information about the environment of the vehicle, the Crash Prediction Network will predict whether a crash occurs or not, while simultaneously, the RL agent would interact with the environment to determine whether a crash really occurred or not. Based on the differences in the output of the two networks, the Crash Prediction Network would be updated to eventually be able to predict crashes with a high level of accuracy.

The operational stage (as show in Fig. 2) of this model is designed such that the inputs as usual are fed to the ML-based component responsible to determine the lane change maneuver to be carried out by the ego vehicle. The vehicle, however, does not act directly on the generated lane change action command. The action command along with the environmental inputs in the form of sensor data are directed to the Crash Prediction Network, which performs its task of predicting the likelihood of a crash. Only if the likelihood is low, is the vehicle allowed to perform the desired actions, else the vehicle is pushed into Fail-safe mode which varies depending on the predicted severity of the crash. It is important to note that for the model to stay relevant to the environment, it needs to learn and improve even in the operational stage. Thus, similar to the training stage the difference between the actual output and predicted output are used to update the model.

Crash Prediction Network is based on Bayesian Deep Learning (BDL). The reason being that other Deep Learning methods in use currently are known to make hard classifications based on what they see and what they perceive. The disadvantage with this method becomes apparent in a system such as an AV where multiple components come together to form a complex whole, an error in one component could have a snowball effect up the pipeline, leading to catastrophic outputs in the later components. A way to get over this problem is to use BDL (McAllister et al. 2017). Bayesian models would provide better results (Kendall and Gal 2017), owing to the fact that such models generate as output a probability distribution with a consideration for uncertainty, which can be exploited for the output regarding the likelihood of a crash that the model is expected to generate. Additionally, it would mean that the model would propagate not only the classification output but also the uncertainty of the model associated with the output, such that the higher-level components can be developed to react in a way that the system behaves conservatively when the uncertainty of the previous components in the pipeline is high.

The proposed system has definite advantages. Most importantly, such a system does not just focus on futuristic autonomous vehicles, but, can even be used in current day Advanced Driving Assistance Systems (ADAS) as well, thereby allowing a smoother transition to Autonomous Vehicles in future. Secondly, the model can be seen as making an intuitively 'informed decision', by taking into consideration data from multiple sources. Additionally, such a system would also generalize and scale well to different scenarios that the vehicle might encounter. One of the major problems that would be encountered during the development of the model, however, is the consideration of handling input data received from different sources in varied formats. Next, redundancy needs to be inbuilt to compensate for sensor failures/malfunctions in such a way that failure of a sensor does not affect the accuracy of the system. Another major aspect, apropos this methodology that needs experimentation and validation is that of having one ML based component supervising another.

Conclusion

This work covered the different aspects of safety for intelligent components which employ machine learning techniques in order to enable the integration of artificial intelligence for autonomous driving. The focus was on the main concerns and challenges to ensure safety in highly critical

applications which are based on machine learning methods, with special emphasis on neural networks. Traditional safety approaches are not sufficiently poised for such systems and therefore, there is a need for more concrete methods like monitoring techniques, such as the one proposed Crash Prediction Network, which guarantees an acceptable level of safety for the system functions. The team is in the process of implementing and evaluating the proposed approach.

References

Bagschik, G.; Menzel, T.; and Maurer, M. 2017. Ontology based scene creation for the development of automated vehicles. *arXiv preprint arXiv:1704.01006*.

Broy, M. 2006. Challenges in automotive software engineering. In *Proceedings of the 28th international conference on Software engineering*, 33–42. ACM.

Burton, S.; Gauerhof, L.; and Heinzemann, C. 2017. Making the case for safety of machine learning in highly automated driving. In *International Conference on Computer Safety, Reliability, and Security*, 5–16. Springer.

Cheng, C.-H.; Diehl, F.; Hinz, G.; Hamza, Y.; Nührenberg, G.; Rickert, M.; Ruess, H.; and Truong-Le, M. 2018. Neural networks for safety-critical applicationsâĂŤchallenges, experiments and perspectives. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, 1005–1006. IEEE.

Committee, S. O.-R. A. V. S., et al. 2014. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J* 3016:1–16.

Cukic, B.; Fuller, E.; Mladenovski, M.; and Yerramalla, S. 2006. *Run-Time Assessment of Neural Network Control Systems*. Boston, MA: Springer US. 257–269.

Darrah, M., and Taylor, B. J. 2006. *Rule Extraction as a Formal Method*. Boston, MA: Springer US. 199–227.

Fagnant, D. J., and Kockelman, K. 2015. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice* 77:167–181.

Fridman, L.; Jenik, B.; and Reimer, B. 2017. Arguing machines: Perceptioncontrol system redundancy and edge case discovery in real-world autonomous driving. *arXiv preprint arXiv:1710.04459*.

Fu, L. 1994. Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics* 24(8):1114–1124.

Fuller, E. J.; Yerramalla, S. K.; and Cukic, B. 2006. Stability properties of neural networks. In *Methods and Procedures for the Verification and Validation of Artificial Neural Networks*. Springer. 97–108.

Gasser, U., and Almeida, V. A. 2017. A layered model for ai governance. *IEEE Internet Computing* 21(6):58–62.

Kalra, N., and Paddock, S. M. 2016. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice* 94:182–193.

Kendall, A., and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Koopman, P., and Wagner, M. 2016. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety* 4(1):15–24.

Liu, Y.; Menzies, T.; and Cukic, B. 2002. Data sniffing-monitoring of machine learning for online adaptive systems. In *Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. 14th IEEE International Conference on*, 16–21. IEEE.

McAllister, R.; Gal, Y.; Kendall, A.; Van Der Wilk, M.; Shah, A.; Cipolla, R.; and Weller, A. V. 2017. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.

Monkhouse, H.; Habli, I.; McDermid, J.; Khastgir, S.; and Dhadyalla, G. 2017. Why functional safety experts worry about automotive systems having increasing autonomy. In *International Workshop on Driver and Driverless Cars: Competition or Coexistence*.

Ray, S. 2010. *Scalable techniques for formal verification*. Springer Science & Business Media.

Salay, R.; Queiroz, R.; and Czarnecki, K. 2017. An analysis of ISO 26262: Using machine learning safely in automotive software. *CoRR* abs/1709.02435.

Seshia, S. A.; Sadigh, D.; and Sastry, S. S. 2016. Towards verified artificial intelligence. *arXiv preprint arXiv:1606.08514*.

Taylor, B. J.; Darrah, M. A.; and Moats, C. D. 2003. Verification and validation of neural networks: a sampling of research in progress. In *Intelligent Computing: Theory and Applications*, volume 5103, 8–17. International Society for Optics and Photonics.

Taylor, B. J. 2006. *Automated Test Generation for Testing Neural Network Systems*. Boston, MA: Springer US. 229–256.

Thrun, S. 1995. Extracting rules from artificial neural networks with distributed representations. In Tesauro, G.; Touretzky, D.; and Leen, T., eds., *Advances in Neural Information Processing Systems (NIPS) 7*. Cambridge, MA: MIT Press.

Yerramalla, S.; Fuller, E.; Mladenovski, M.; and Cukic, B. 2003. Lyapunov analysis of neural network stability in an adaptive flight control system. In *Symposium on Self-Stabilizing Systems*, 77–92. Springer.

Zilke, J. R.; Mencía, E. L.; and Janssen, F. 2016. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, 457–473. Springer.