Towards safety monitoring of ML-based perception tasks of autonomous systems

Raul S. Ferreira
LAAS-CNRS, Universite de Toulouse, France
rsenaferre@laas.fr

***Abstract*—Machine learning (ML) provides no guarantee of safe operation in safety-critical systems such as autonomous vehicles. ML decisions are based on data that tends to represent a partial and imprecise knowledge of the environment. Such probabilistic models can output wrong decisions even with 99% of confidence, potentially leading to catastrophic consequences. Moreover, modern ML algorithms such as deep neural networks (DNN) have a high level of uncertainty in their decisions, and their outcomes are not easily explainable. Therefore, a fault tolerance mechanism, such as a safety monitor (SM), should be applied to guarantee the property correctness of these systems. However, applying an SM for ML components can be complex in terms of detection and reaction. Thus, aiming at dealing with this challenging task, this work presents a benchmark architecture for testing ML components with SM, and the current work for dealing with specific ML threats. We also highlight the main issues regarding monitoring ML in safety-critical environments.**

***Index Terms*—Dependability, machine learning, autonomous vehicles**

I. Introduction

ML is widely applied for performing important autonomous tasks in safety-critical domains such as autonomous vehicles or robotics. Such tasks include perception, navigation, and control. The reason is the flexibility of using ML in unmodeled environments, the increasing amount of labeled data, and computational power. However, an ML model takes decisions based on trained past data. This ML model is validated by analyzing specific metrics, for instance, the discrepancy between the predicted values and the true answers (labels). If these results are satisfactory, it goes to production. However, the adoption of ML for safety-critical systems can be hindered by some important drawbacks of these algorithms.

First, these algorithms are based on data, and the available data tends to be incomplete regarding the environment complexity, classes, and corner cases existent in real-world data. Second, an ML model tends to learn incomplete relationships on this data. That is, it is not capable of correctly generalizing real-world data just using the data available during training. Third, due to the aforementioned limitations on data and in the ML model, it can wrongly predict a class even with 100% of confidence [1]. Finally, these decisions are hard to interpret, especially modern ML algorithms such as DNNs, making them a black-box software. These drawbacks are challenging for verification and monitoring of ML.

The ML community researches for decades on how to increase the robustness of these algorithms by making them detecting and adapting to the different ML threats at runtime. There are several examples applied to novelty classes [2], anomalies [3], distributional-shift (concept-drift) [4], adversarial examples [5], and so on. However, despite the huge advances in the field, such algorithms cannot avoid the fact that even performing correctly 99% of the time they still cannot guarantee safety. That is, this high-performance algorithm will fail 1% of the time during operation, and it is enough to lead to several and serious hazards. This highlights, even more, the importance of redundant techniques for monitoring these algorithms during operation (runtime).

A possible approach for monitoring ML decisions at runtime is applying a safety monitor (SM) [6] as a part of the runtime verification mechanism. This fault-tolerance approach is widespread in safety-critical domains [7]. The SM is responsible for maintaining the system in a safe state despite the occurrence of hazardous situations. It is usually composed of a detection mechanism and a recovery system. However, for complex applications where ML is required to provide solutions, designing an SM can be intractable due to the need to verify millions or even billions of parameters generated by the ML model. Besides, SM might face other issues such as to define detection mechanisms for this ML model. Therefore, the focus of this work is to develop efficient SM for ML components. In this thesis, we adopt the dependability vocabulary (threats, faults...) [7] applied to ML issues.

This research tries to answer four research questions (RQ):

1) What type of ML threats can be detected?
2) How to monitor ML threats at runtime?
3) How to benchmark different runtime monitors?
4) How to intervene after the detection?

This thesis started in March 2019 and is both conceptual and experimental. It includes two industrial collaboration involving the study of runtime monitoring perception tasks for autonomous vehicles (AV). Therefore, in Section II we highlight the current challenges linked to the aforementioned RQs. In Section III, we present current related works to address these challenges. In Section IV, we propose possible solutions that deal with the drawbacks of the previous methods. Finally, in Section V, we conclude the paper and describe the work plan.

II. Challenges

In this section, we frame the research domain and explain the challenges for answering each RQ mentioned earlier.

*1) Identification of ML threats for specific tasks (**RQ1**):* RQ1 focuses on identifying the domain, its specific tasks along with the observable data, and identify the possible threats. ML threats can be observed during two different steps in the software life cycle: during development or interaction. Development faults happen during the design and training of the ML algorithm, while interaction faults happen during runtime. Possible threats during the design phase include security vulnerabilities, hidden bugs, bad model maintenance, specification mismatch, data incompleteness, white noise, bad requirements engineering, inadequate ML development, and insufficient test coverage. Regarding the runtime phase, possible threats include distributional shift, anomalies, novel classes, adversarial inputs, and noisy inputs. Besides, threats are also dependent on the domain and its tasks, along with the observable data applied to this task.

*2) Identification of runtime monitoring approaches (**RQ2**):* Techniques for detecting a hazardous situation based on functional specifications are usually applied for traditional systems and some of them [8], [9] can be applied to monitor specific ML threats. On the other hand, the use of techniques based on data became very popular, such as pattern classification [2], feature learning [10], neuron matching [11], instance-to-distribution verification [12], distribution-to-distribution verification [13] and so on. Reacting to an ML decision depends on its ability to detect and identify possible threats in the ML component. Thus, RQ2 focuses on understanding and mapping which approaches can be applied for monitoring threats for a perception task. Moreover, RQ2 aims at applying and combining these approaches while minimizing their drawbacks.

*3) A framework for benchmarking SM (**RQ3**):* RQ3 focuses on constructing a benchmark architecture following standard guidelines. There is extensive literature regarding how to perform dependability benchmarks for traditional systems, and for benchmarking ML algorithms. However, there is a lack in the literature on how to benchmark a system which both work together in the same system. There is also a lack of formalism regarding which metrics should be considered when testing fault tolerance mechanisms for ML models. Besides, determine which methods should be used as baselines, and upper bound, for these experiments is another topic not well-defined in the literature. Finally, such a framework for benchmarking SM needs to allow that the fault templates for each ML threat are parameterized, and the possibility of combining mutated datasets. Therefore, RQ3 aims at researching these important methodological challenges and proposing an experimental architecture and methodology.

*4) Modeling reactions according to specific threats (**RQ4**):* Another challenging task for SM is to determine which strategies to use for better reacting when a threat is detected. The reaction is as important as the detection, and the RQ4 is focused on finding new ways to do it since traditional methods for generating safety rules tend to be infeasible due to the complexity of the ML algorithm and the scenario [14]. Besides, several reactions can be applied in the perception module (e.g. changing the ML decision) or in the control module (e.g. decelerating the car). It poses another challenge about which intervention is better to use, in which part of the system should be applied to this intervention, and so on.

III. Related works

SM for ML components is a technique with a growing interest in the autonomous systems field, and the current literature proposes two different approaches to do it:

A. SM based on a functional specification

This approach is generally based on a physical model of the system or the environment, and on properties they should guarantee. Some techniques use a rule-based approach for verifying the safety of ML components, considering this component as a **black-box**. For example, safety rules can be applied to verify if a vehicle can completely stop before reaching an obstacle ahead [9], or to trigger an intervention for avoiding a collision when an ego vehicle is approaching a leading vehicle [14]. In both cases, external (exteroceptive) sensors, such as distance sensors, are observed along with some internal (proprioceptive) sensors, such as speed, in order to evaluate if a safety property will be violated. Such techniques have the advantage of automatically generate safety strategies [6]. However, analyzing the ML as a black box can be considered as a drawback since accessing a certain level of properties that lead an ML to give a particular decision is important for SM in complex scenarios. Besides, these techniques tend to be infeasible in a complex scenario [14].

Another model-based technique is the use of code assertions, also called model assertions [8]. This technique is an adaptation of the classical program assertions as a way to monitor and improve ML models. The idea is to verify properties in the program logic in order to point possible failures. For example, monitoring temporal stability during operation by verifying if the DNN output regarding an object is flickering in and flickering out in the camera, which indicates a possible failure. Even these techniques can be applied during design and operation, it alone cannot guarantee that a DNN decision is safe. The reason is that, for some corner cases, ML outputs wrong decisions that cannot be verified by inspecting the code logic or the sensor values and lead to hazards. Such corner cases come from regions of uncertainty [15].

B. SM based on training data

This approach is usually purely based on data instead of using a physical model for building the solution. For example, one approach is to monitor the **neuron patterns** observed from the layers of the DNN. For instance, an SM can monitor the values in the last layer of a DNN [16], verifying the confidence level of the decision. However, it is not reliable since DNNs can output a wrong decision even with a high level of confidence. Cheng et al [17] suggest comparing the recorded patterns of the DNN activation functions during the training with the ones related to the inputs during runtime. After the standard training process, a runtime monitor is created by feeding the training data to the network again

in order to store the neuron on-off activation patterns using binary decision diagrams (BDD). During runtime, the monitor rejects the decision made by the network if the current current BDD pattern is not similar to the ones stored in the monitor during the training. One of the advantages is that it partially addresses the novelty class problem by signaling variations in the patterns of the DNN activation functions. However, choosing a good threshold for determining which variation could be signaled as a novel class can be difficult, varying between the datasets. Besides, the memory required for constructing the BDD grows quickly in respect of the DNN's size.

To overcome the aforementioned problems Henzinger et al [11] propose to verify neuron activation patterns values but using a 2D projection instead of a BDD. This projection is an abstraction box built by taking the maximum and minimum values of activation function values during training. During runtime, it inspects if the output of an activation function, after a new input pass through it, falls inside of this abstraction box. If not, it raises an alarm considering this input as a novel input. By doing that, the authors reduced the memory issue and speed up the construction and detection. However, 2D projections tend to loose information when applied to high dimensional data, impacting the accuracy of the detection. Moreover, applying linear projections to nonlinear activation functions tends to impact negatively in the detection performance.

Another approach is to monitor the DNN **inputs** based on a radius distance threshold calibrated during the training [18]. The idea is to perturb the DNN inputs, observe the true answer, and determining how large is the distance regarding the DNN decisions. By doing that, the authors observed that the radius distance of a DNN that correctly classified the perturbed inputs are much larger than that ones that misclassified such inputs after a perturbation. The authors uses this intuition to protect the DNN decisions from adversarial attacks. The advantage of this approach is that it does not need to inspect the internals of the DNN. The drawback is that it tends to be biased to the data used during the training. In the next section, we propose some directions for answering RQs in Section I while overcoming the challenges in Section II.

IV. Proposed methods

For RQ1, we chose to focus on perception tasks in the domain of autonomous vehicles (AV) due to the importance of this task in such a safety-critical domain. The observable data for this task is mostly collected from exteroceptive sensors such as cameras, and LiDAR [19], and the outputs and intermediate values of ML algorithms. Hence, in this thesis, we focus on the runtime monitoring of ML components when performing image classification tasks in AV. For RQ2, we chose to build a combined approach using techniques both from functional specification and based on data. This approach is capable of inspecting not just the observable parameters of the DNN that impact on its decision such as input features, neuron patterns, but also other properties of the system. The reason is that the ability to detect errors or unsafe behaviors before a hazardous condition is connected to the perception of a safety model. Thus, a safety model needs to be able to extract fine-grained information from the ML component to increase its perception ability.

Regarding RQ3, this thesis compares different SM techniques by using an experimental framework based on the FARM [20] methodology. This methodology is composed of a fault load for creating the mutated datasets according to a specific ML threat; a workload specific for the domain; readouts, or raw outputs, generated from the tests; and measurements applied on these readouts. Hence, as illustrated in Figure 1, the final architecture for benchmark each SM for a specific ML threat is divided into three modules: dataset generation, system testing, and evaluation. That is, the dataset generation

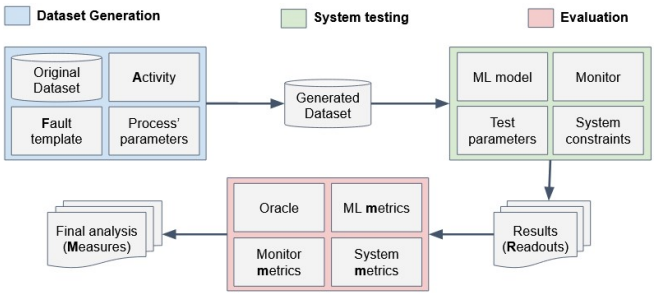

Fig. 1. Benchmark architecture.

module produces the mutated datasets containing ML threats, for example, adversarial images. After it, the system testing module is responsible for performing tests using the ML-based along with the SM on these generated datasets. Finally, the evaluation module produces the final evaluation of the system using the outputs produced by the tests.

For RQ4, there are promising alternatives for reacting when a detection is made. For example, using a modified simplex architecture [21] with two controllers: a high-performance and a verified-safe controller. In this architecture, the first controller uses ML, while the second one is based on a safety model. When the decisions of the first controller are potentially unsafe within restricted conditions, the control goes to the second controller. In this architecture, there is an adaptation module responsible for updating the high-performance controller and make it reliable against previously seen unsafe decisions. Hence, it creates a reverse switching mechanism that is able to give control back to the high-performance controller.

Another alternative is to synthesize safety rules just for critical counterexamples [15]. That is, inputs that can lead to a failure of an entire system when the ML gives a wrong decision (e.g. an ML component *wrongly* missed an obstacle, giving no time to react and to avoid a crash). By doing that, it is also possible to find the inputs that are only safe if the ML decision is correct (e.g. an ML component *correctly* identifies an obstacle with sufficient time to react and to avoid a crash).

V. Conclusions and work plan

The use of ML by the automotive industry is quickly increasing due to its flexibility to learn from data and its ability to perform complex tasks such as perception. Guaranteeing that these autonomous tasks output correct and reliable decisions are an open challenge in the dependability and ML fields. Since each ML threat has a specific characteristic, we believe that an ensemble of specialized SM could achieve better results than a single SM.

This research aims at contributing to the field of dependability by giving robust research directions on how to monitor the main threats that an ML-based perception system can face at runtime. Despite several works on algorithm robustness, there are just a few works about runtime monitoring ML, which highlights the importance of this research. Another expected contribution is to provide research directions regarding the methodology of benchmarking SM along with ML, a subject not well-explored in the literature.

This three-year thesis started in March 2019, and it has the following work plan:

1st year: Literature review and taxonomy building

- What are the major threats for trusting in ML, from preprocessing up to production, and maintenance?
- What are the dependability means applied for monitoring traditional systems, and ML-based systems? What are the differences between them?
- What are the current solutions for monitoring ML tasks in a safety-critical domain (autonomous vehicles)?
- Preliminary taxonomy on approaches for detecting ML threats at runtime.

2nd year: Benchmarking

- Developing and selecting a set of SM candidates.
- Experiments comparing our SM for novelty class detection with related works.
- Secondment at Fraunhofer (DE) institute to combine techniques of uncertainty estimation to our SM.
- Experiments comparing our SM for detection of distributional shift with related works.
- Experiments comparing our SM for detection of adversarial inputs with related works.

3rd year: Consolidating the proposal

- Secondment at Jaguar/Land Rover (UK) for developing and testing a combination of SM in a more realistic scenario.
- Experiments comparing our combined SM with related works in a scenario with multiple types of threats.
- Finalization of the generic framework to monitor and handle safety of autonomous systems.

Acknowledgements

This project has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020. Grant Agreement: 812,788. The author thanks Jérémie Guiochet, Hélène Waeselynck, Mario Trapp, and Harita Joshi for their guidance during this PhD's 1st year.

References

[1] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International conference on machine learning (ICML), New York, United States*, 2016, pp. 1050–1059.

[2] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[3] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft, "Image anomaly detection with generative adversarial networks," in *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 2018, pp. 3–17.

[4] R. S. Ferreira, G. Zimbrão, and L. G. Alvim, "Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency," *Information Sciences*, vol. 488, pp. 219–237, 2019.

[5] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[6] M. Machin, J. Guiochet, H. Waeselynck, J.-P. Blanquart, M. Roy, and L. Masson, "SMOF: A safety monitoring framework for autonomous systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 5, pp. 702–715, 2018.

[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[8] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, "Model assertions for monitoring and improving ML model," *arXiv preprint arXiv:2003.01668*, 2020.

[9] U. Ozguner, C. Stiller, and K. Redmill, "Systems for safety and autonomous behavior in cars: The darpa grand challenge experience," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 397–412, 2007.

[10] A. Gupta and L. Carlone, "Online monitoring for neural network based monocular pedestrian pose estimation," *arXiv preprint arXiv:2005.05451*, 2020.

[11] T. A. Henzinger, A. Lukina, and C. Schilling, "Outside the box: Abstraction-based monitoring of neural networks," *arXiv preprint arXiv:1911.09032*, 2019.

[12] S. Matiz and K. E. Barner, "Inductive conformal predictor for convolutional neural networks: Applications to active learning for image classification," *Pattern Recognition*, vol. 90, pp. 172–182, 2019.

[13] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6044–6053.

[14] F. Al-Khoury, "Safety of machine learning systems in autonomous driving," Master's thesis, KTH Royal Institute of Technology School of Industrial Engineering and Management, Stockholm, Sweden, 2017.

[15] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," *Journal of Automated Reasoning*, vol. 63, no. 4, pp. 1031–1053, 2019.

[16] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *CoRR*, vol. abs/1610.02136, 2016. [Online]. Available: http://arxiv.org/abs/1610.02136

[17] C.-H. Cheng, G. Nührenberg, and H. Yasuoka, "Runtime monitoring neuron activation patterns," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy*. IEEE, 2019, pp. 300–303.

[18] J. Liu, L. Chen, A. Mine, and J. Wang, "Input validation for neural networks via runtime local robustness verification," *arXiv preprint arXiv:2002.03339*, 2020.

[19] X. Meng, H. Wang, and B. Liu, "A robust vehicle localization approach based on gnss/imu/dmi/lidar sensor fusion for autonomous vehicles," *Sensors*, vol. 17, no. 9, p. 2140, 2017.

[20] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins, and D. Powell, "Fault injection for dependability validation: A methodology and some applications," *IEEE Transactions on software engineering*, vol. 16, no. 2, pp. 166–182, 1990.

[21] D. Phan, N. Paoletti, R. Grosu, N. Jansen, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," 2019.