ANITI
ARTIFICIAL & NATURAL INTELLIGENCE TOULOUSE INSTITUTE

Postdoctoral Proposal in Artificial Intelligence and Machine Learning

Runtime Verification for Critical Machine Learning Applications

Advisor (s):
Kevin Delmas, kevin.delmas@onera.fr, https://www.onera.fr/en/staff/kevin-delmas
Jérémie Guiochet, jeremie.guiochet@laas.fr, http://homepages.laas.fr/guiochet

Net salary: Negotiable with a minimum of 2600€ per month with some teaching (20 hours per year on average)

Duration: one year (renewable once)

Description

In the last decade, the application of Machine Learning (ML) has encountered an increasing interest in various applicative domains, especially for a wide panel of complex tasks (e.g. image-based pedestrian detection) classically performed by human operators (e.g. the driver). In ML, the objective is to synthesize an intended function (e.g., detect a pedestrian on an image) through a set of examples (images of road). The massive usage of such techniques has demonstrated its effectiveness way beyond other classical methods. Obviously, the designers of critical systems would like to benefit from the effectiveness of ML-based models mainly for complex image processing and model reduction. But, above effectiveness, the designers of critical systems must demonstrate that the obtained models are reasonably safe. Providing elements demonstrating the safety of a system is a classical issue addressed by various techniques tailored to the nature of the system, and covered by many safety standards (DO178C in aeronautics, ISO26262 in automative, IEC61508 in electronics, etc.). Nevertheless, the specificities of ML-based software disclose new safety threats that are not addressed by classical techniques. Despite very good results during training and testing of a ML software, it is not possible to provide sufficient guarantees that the training data set would be sufficient for expected real life situations, that during operational life the system may not face adversary situations (situation slightly different from training ones, but which lead to a complete different result of the ML, also called adversaries attacks), or that the distribution of situations may be different from the ones during training (distribution shift). All these threats are a major brake to ML deployment in safety critical applications.

Most of works are focusing on the training data quality in order to increase robustness of the ML algorithm. However, to avoid overfitting, it is accepted that developing the dataset is limited, and a promising approach is to monitor the system at runtime, during operational life, in order to keep the system in a safe state, despite errors of the ML. A first approach inspired from fault tolerance and close to safety monitoring [3], is to adapt the simplex architecture to the monitoring of a neural controller, using a decision block able to detect an error and to switch from the neural controller to a high assurance controller (but less performant) [4]. Some works as [1] may be used to monitor the distance of input/ouput distribution during the exploitation vs the one observed during the learning phase and raise an alert when a "significant gap" is observed. Other works like [2], dedicated to Neural Networks, propose to collect neuron activation patterns during the learning phase and, thanks to an online monitoring, detect the occurrence of an unknown activation pattern that may indicate an erroneous prediction.

All these works are ongoing, with very preliminary results, and actually no safety model is integrated in such proposals. We propose in this post-doc to specify, implement and verify a new runtime verification approach for ML, an adversarial runtime monitor. This approach is based on adversaries generated at runtime, and used to assess if the ML maybe fooled in an unsafe state. This might lead the monitor to detect if the ML is in a potential unsafe erroneous state, or in a potential erroneous state but safe. Once such a monitor would be designed, we also plan to use formal methods (verification) to prove the correctness of the monitor. This work will be applied to a case study, a ML software for drone collision avoidance studied and deployed in the context of the Delta project[1].

[1] the code of the Delta project is available at https://github.com/delta-onera/delta_tb/tree/master/workspace/isprs_

References

[1] Mahdieh Abbasi, Arezoo Rajabi, Azadeh Mozafari, Rakesh B. Bobba, and Christian Gagné. Controlling over-generalization and its effect on adversarial examples generation and detection. *ArXiv e-prints*, 1808.08282, 8 2018.

[2] Chih-Hong Cheng, Georg Nührenberg, and Hirotoshi Yasuoka. Runtime monitoring neuron activation patterns. *CoRR*, abs/1809.06573, 2018.

[3] Mathilde Machin, Jérémie Guiochet, Hélène Waeselynck, Jean-Paul Blanquart, Matthieu Roy, and Lola Masson. SMOF - A Safety MOnitoring Framework for Autonomous Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):702–715, May 2018.

[4] Dung Phan, Nicola Paoletti, Radu Grosu, Nils Jansen, Scott A. Smolka, and Scott D. Stoller. Neural simplex architecture. *ArXiv e-prints*, abs/1908.00528, 2019.