Proc. of the 9th Int. Real-Time Scheduling Open Problems Seminar (RTSOPS 2018), Barcelona, Spain, July 3, 2018.

Deep Neural Networks for Safety-Critical Applications: Vision and Open Problems

Daniel Casini, Alessandro Biondi, Giorgio Buttazzo
Scuola Superiore Sant'Anna
Pisa, Italy
Email: {daniel.casini, alessandro.biondi, giorgio.buttazzo}@santannapisa.it

I. Introduction

Autonomously-driving cars may become part of our everyday lives in a few years. One of the key factors that will likely enable the development of autonomous systems is artificial intelligence and, in particular, deep neural networks (DNNs). According to the results of a popular challenge for assessing the performance of DNNs [1], the error rate in image classification reduced from 28% in 2010 to 2.3% in 2017 [2], surpassing the human capability, assessed between 5% and 10% (see Figure 1). Neural networks are computing system inspired by the structure of mammalian brains, composed of many neurons (represented by nodes) exchanging signals through synaptic channels (represented by weighted edges). Synaptic weights are tuned during a supervised learning procedure, in which the network is trained by examples to match a set of expected outputs when specific inputs are presented. The adoption of neural networks in safety-critical scenarios presents many issues. For instance, the difficulty of understanding the meaning of weights configurations and the difficulty of predicting the network output to similar input patterns represents a crucial problem for certification: who would certificate something whose behavior is not completely understandable at design time and cannot be predictably replicated? A possible solution may consist in adopting algorithms based on *hard computing* (i.e., with a well-defined behavior) aimed at acting as *supervisors* for DNNs. The supervisor would detect misbehaviors from the neural network, redirecting the actuation to some defined safe actions. A further improvement (but still in terms of average-case behavior) can be achieved using redundant neural networks in conjunction for performing the same task, and merging the result by means of a voting mechanism.

The complexity of a neural network also makes it prone to attacks and malfunctionings, representing a security threat: what if an attacker exploits the weakness of a DNN to take control over the related, safety-critical, steering system? Morever, DNNs are typically executed on top of general purpose operating systems, whereas actuations take place in a Real-Time Operating System (RTOS). For these reasons, guaranteeing isolation between DNNs and safety-critical tasks represents an important issue.

A popular and effective technique to achieve isolation using a single heterogeneous multi-processor platform is to use a *hypervisor* to separate a computationally intensive processing (i.e., DNNs) and safety-critical activities (e.g., AUTOSAR compliant) in different domains. In this way, the two domains can safely co-exist, also adopting different operating systems. This solution is shown in Figure 2.

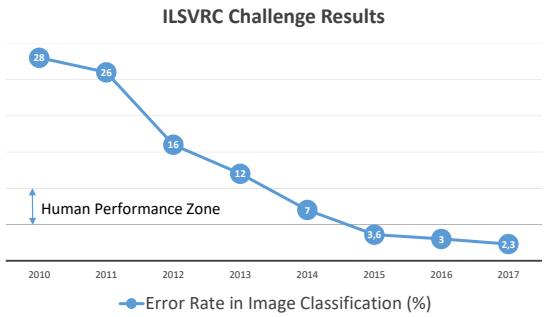

Fig. 1. Evolution of DNNs over the years in terms of error rate in image classification tasks.

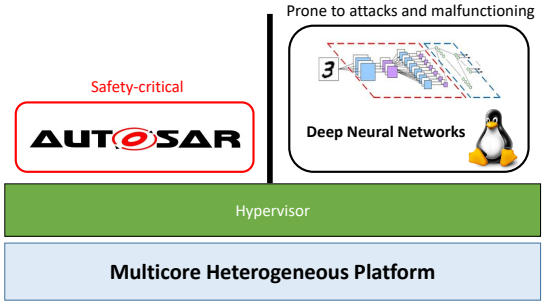

Fig. 2. Sample architecture for achieving isolation between DNNs and safety critical tasks when the underlying hardware platform is shared.

Another key issue is represented by the temporal predictability of DNNs. However, predictability may be a requirement only when DNNs are used in their *inference phase* (i.e., the network is already trained and it is only queried). Although a DNN workload can be represented with a direct acyclic graph[1] (DAG), finding a representative task model (e.g., that accounts

[1]In its classical form: for instance the Tensorflow programming model provides some extensions for branching and looping, which leads to different task models [3].

for memory accesses due to tensors exchanged between directly connected nodes) is fundamental to correctly characterize its temporal behavior. Deep Neural Networks execute on hardware platforms composed of heterogeneous processing units, such as (possibly asymmetric) multi-core CPUs, GPUs and portions of reconfigurable FPGAs. Recently, ad-hoc application specific integrated circuits (ASICs) have been also produced and may be commonly included in commercial heterogeneous platforms soon. The tensor processing unit (TPU) [4] produced by Google and compatible with Tensorflow, one of the most used frameworks for developing Deep Neural Networks, is a notable and promising example. Accounting for the underlying hardware architecture is unavoidable when designing more tailored analysis techniques, but not always simple. This is the case of NVIDIA GPGPUs, whose internal structure and mechanisms are not publicly documented. For these platforms, research can only rely on black-box characterizations based on experimentation results [5]. The support of FPGAs is not yet available for most of the DNNs frameworks (e.g., Tensorflow), but efforts have been done to make them supported [6]. An interesting area of research may also evaluate the execution of DNNs with dynamic partial reconfiguration [7], which recently have been made available (as a prototype) under Linux [8]. Using this approach, computationally intensive nodes may be accelerated by executing them on a FPGAs, exploiting dynamic partial reconfiguration to avoid having a fixed binding between a portion of FPGA area and one or more DNN operations. The support of heterogeneous platforms opens to research on partitioning neural network nodes to different types of devices. For instance, the placement algorithm (the one aimed at deciding where each node should be executed, e.g., in a GPU or in a CPU) still needs to be improved in TensorFlow [3]. Among the usual frameworks for developing and executing DNNs (e.g., Caffe [9], Tensorflow [10], Torch [11], etc.), novel frameworks specialized for inferencing DNNs have been recently developed. For instance, it is worth mentioning TensorFlow Serving [12], an inference framework aimed at increasing the throughput of a DNN by grouping individual inference requests into batches for joint execution on a GPU, and NVIDIA TensorRT [13], which implements a pre-processing phase aimed at optimizing the neural network for a future inference in an NVIDIA GPU. In this way, latency can be decreased and throughput increased. Predictability of optimized inference engines for DNNs should also be evaluated as a future research work.

II. Open Problems: Categorization and Key Questions

A. Certification

1) How to support DNNs in a safety-critical context to build a system that is prone for being certified?
2) Which type of *supervision algorithms* are needed to detect wrong outputs and redirect actuation to safe actions?

B. Isolation

1) How to avoid that the complexity of DNNs may lead to security threats for a safety-critical system running on top of a shared hardware platform?
2) Which mechanism have to be provided to allow them interacting while running in different operating systems?

C. Predictability in the execution

1) Which is a suitable task model to describe and analyze the temporal behavior of a DNN?
2) How to account for novel (highly heterogeneous) computing platforms?
3) How to account for inference engines that affect the DNN's execution?

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[2] [Online]. Available: http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Man, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vigas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2015. [Online]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf

[4] [Online]. Available: https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

[5] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "GPU scheduling on the NVIDIA TX2: Hidden details revealed," in *Proceedings of the 38th IEEE Real-Time Systems Symposium (RTSS 2017)*, Paris, France, December 5-8 2017.

[6] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, "Fp-dnn: An automated framework for mapping deep neural networks onto fpgas with rtl-hls hybrid templates," in *Field-Programmable Custom Computing Machines (FCCM), 2017 IEEE 25th Annual International Symposium on*. IEEE, 2017, pp. 152–159.

[7] A. Biondi, A. Balsini, M. Pagani, E. Rossi, M. Marinoni, and G. Buttazzo, "A framework for supporting real-time applications on dynamic reconfigurable FPGAs," in *Real-Time Systems Symposium (RTSS), 2016 IEEE*, Porto, Portugal, November 29 - December 2 2016.

[8] M. Pagani, A. Balsini, A. Biondi, M. Marinoni, and G. Buttazzo, "A linux-based support for developing real-time applications on heterogeneous platforms with dynamic fpga reconfiguration," in *System-on-Chip Conference (SOCC), 2017 30th IEEE International*. IEEE, 2017, pp. 96–101.

[9] [Online]. Available: http://torch.ch/

[10] [Online]. Available: https://www.tensorflow.org/

[11] [Online]. Available: http://caffe.berkeleyvision.org/

[12] [Online]. Available: https://github.com/tensorflow/serving

[13] [Online]. Available: https://developer.nvidia.com/tensorrt