Preprint: To Appear in IEEE Micro Special Issue on Cognitive Architectures, 2017

Cognitive Computing Safety: The New Horizon for Reliability

Yuhao Zhu
yzhu@utexas.edu
The University of Texas at Austin

Vijay Janapa Reddi
vj@ece.utexas.edu
The University of Texas at Austin

1 Introduction

Recent advances in cognitive computing have brought widespread excitement for a variety of machine learning-based intelligent services, ranging from autonomous vehicles to smart traffic light systems. To push such cognitive services more toward reality, recent research has focused extensively on improving the performance, energy-efficiency, privacy, and security of cognitive computing platforms.

Among all the issues, a rapidly rising and critical challenge to address is the practice of *safe cognitive computing*—, i.e., how can we architect machine learning-based systems to be robust against uncertainty and failure to guarantee that they perform as intended without causing harmful behavior. Addressing the safety issue will involve close collaboration among different computing communities, and we believe *computer architects must play a key role*. In this position paper, we first discuss the meaning of safety and the severe implications of the safety issue in cognitive computing. We then provide a framework to reason about safety, based on which we outline several opportunities for the architecture community to help make cognitive computing safer.

2 Safety in Cognitive Computing Systems

Safety in cognitive computing is inherently associated with the accuracy issue in machine learning. Just as with any system-level behavior, accuracy can be classified into two categories: average-case accuracy and worst-case accuracy. Worst-case accuracy determines the safety of a cognitive computing system. Improving the worst-case accuracy, however, is hard due to the lack of interpretability in machine learning. Machine learning algorithms are inherently stochastic approximations of the ground truth.

Meaning of Safety Most machine learning systems today focus on the average-case accuracy. In mature machine learning based application domains, such as image classification, the average accuracy of well-trained machine learning models could be 99%.

However, just as data center systems suffer from the long tail latency issue [7], machine-learning systems suffer from "tail accuracy" where a small fraction of requests exhibit extremely poor accuracy due to uncertainties in the machine learning models and form a long accuracy tail distribution that is hard to curtail.

Tail accuracy leads to poor worst-case accuracy guarantees. In mission critical systems, worst-case accuracy is known to raise serious safety concerns. For example, the autopilot system in a self-driving car might be working "ideally" for millions of miles, but it only takes one life-threatening incident to cause significant distrust in such autonomous systems [14].

Worst-case Accuracy Tolerating worst-case accuracy is not an issue that is unique to the cognitive computing domain. For instance, in the aviation industry, engineers constantly address the issue of safety. But there is a notable difference between cognitive systems and today's prevalent aviation systems that make it fundamentally hard to guarantee safety in cognitive computing—the *system interpretability.*

System interpretability means the ability to rationalize and identify the root-cause of a system failure. From time to time, the aviation industry suffers from mid-air tragedies, but for every one of such tragedies, investigators can understand what exactly went wrong, how to fix it, and most importantly, how to prevent such incidents from happening again in future. This is because flight control is based on control theory, physics, and aerodynamics that are interpretable. For example, in the famous Air France Flight 447 incident, the investigators were able to ac-

curately attribute the crash to an aerodynamic stall, which was then attributed to the mishandling of the pitot tubes being obstructed by ice crystals. The whole accident was a pivot point for the civil aviation industry, leading to an overhaul of how measurement devices are installed and how pilots are trained to handle instrument anomalies [2].

In contrast, the same ability to root-cause an incident and learn from it cannot be said for cognitive systems because, at least at present, we lack a deep understanding of how and why machine learning model works in the first place. The lack of system interpretability will only worsen as we begin to compose many such systems together, effectively increasing the opaqueness of the system. If we cannot explain how each component works, we can not attribute the root cause of a whole system failure to a particular component. For example, it is unclear how we would improve a self-driving car, which is a multi-component system involving several decision-making stages, to prevent even the same incident that led to a particular crash from recurring.

3 Bridging the Safety Gap

Safety in cognitive computing systems is a multi-disciplinary problem. However, computer architects are no stranger to building safe systems. We commonly refer to it as reliability. For decades, the computer architecture community has managed to build highly resilient and fault-tolerant architectures. For instance, we guardband, or allocate a large operating margin, to tolerate process, thermal, and voltage (PVT) variations [4]. We have also established fundamental redundancy-based techniques, such as triple module redundancy (TMR) and instruction duplication, to detect and recover from errors [13].

Resilient architectures and safe cognitive systems share a similar goal: guaranteeing expected system behaviors in the event of a failure. Therefore, architects have a unique opportunity to transfer the experience of building resilient architectures to building safe cognitive computing architectures, and to design new solutions to overcome new safety challenges. To that end, we introduce the notion of "safety-gap"—a framework for computer architects to reason about the safety issue in cognitive computing, based on which we outline several research opportunities.

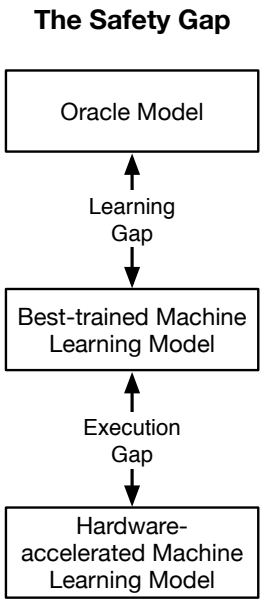

Figure 1: The safety gap in cognitive computing.

The Safety Gap The safety gap of a cognitive system refers to the gap between the worst-case accuracy guarantee that an oracle system demands and the actual accuracy that a particular implementation provides. The safety gap is the composition of two gaps: the learning gap and the execution gap (Fig. 1).

The "learning gap" refers to the gap between the oracle and the best-trained machine learning model. The learning gap exists because even the best machine learning model is likely not a perfect representation of the objective reality and thus introduces error in certain scenarios. For instance, an image classifier mistakenly recognizing a cat in an image as a dog is the result of the learning gap. The learning gap is typically introduced during the model training stage.

Using reliability lingo, errors introduced by the learning gap can be regarded as a form of permanent faults (as opposed to transient faults) caused by "design bugs." Therefore, they are not amenable to traditional backward error recovery techniques (e.g., checkpointing) as well as forward error recovery techniques (e.g., TMR or execution duplication [5]).

The "execution gap", on the other hand, is introduced during the model execution (inference) stage. The execution gap is introduced in two forms. First, to improve the performance and energy-efficiency of model inference, hardware architects often build spe-

cialized accelerators that rely on various optimizations, such as weight compression, data quantization, and SRAM fault injection [12, 10]. These optimizations are "unsafe" because they intentionally trade-off accuracy with execution efficiency. Second, model execution could also suffer from traditional reliability emergencies such as memory failures, nondeterminism, and real-time violations [9]. Overall, hardware execution introduces additional sources of error, exacerbating the worst-case accuracy.

To improve the safety of cognitive computing, we believe that the objectives for computer architects are two-fold. First, we must ensure that in building hardware accelerators, we do not introduce an execution gap while still providing the performance and energy benefits of hardware specialization. Second, we must provide vehicles that help improve model learning accuracy and minimize the learning gap.

To that end, we discuss a few potential directions under both objectives to foster research in this emerging and important topic. Broadly, our suggestions involve tools, architecture-level design and principles that dictate accountable system operation:

Avoiding the Execution Gap

- *Define safety semantics*. In fault-tolerant systems, failure semantics are well-established [6]. Failure semantics dictate how a system degrades gracefully and in what expected manner in the event of a failure. We need to define a similar set of safety semantics at the architecture layer for systems containing machine learning accelerators. The semantics need to specify which safety violations are tolerable and intolerable, and the semantics should also specify how the system degrades during a failure.

- *Build a safe whole out of less safe parts*. Although one accelerator might introduce the execution gap, an ensemble of them might not [8]. One promising approach is to build a cluster of accelerators where some implement simple models that are interpretable but provide less average-case accuracy (e.g., linear regression, decision tree) while others implement complex models that are difficult to interpret but more accurate on average (e.g., neural networks). Such a system improves the worst-case accuracy by routing requests to the simple models when they can be interpreted as safe on simple models.

- *Combine control theory with machine learning*. It is promising to leverage machine learning to take care of average-case accuracy while relying on control theory principles for worst-case accuracy [3]. Computer architects already started using control theory to optimize various architectural metrics such as performance and power [11]. Accuracy is a natural next step.

Minimizing the Learning Gap

- *Co-design machine-learning algorithms with hardware*. Accelerating machine learning algorithms would let algorithm people quickly iterate ideas and speedup the process of closing the learning gap. The key is to co-design the algorithm and the hardware. While most architecture research so far has been primarily focused on convolutional neural networks (CNN), more focus should be dedicated to understanding start-of-the-art algorithms and learning techniques beyond just CNNs [1].

- *Profiling and debugging support*. In aviation systems, flight data recorder and cockpit voice recorder are critical in assisting investigators to pinpoint the root-cause of an air crash. Similarly, we need to provide extensive profiling and debugging support to help system designers understand instances where the cognitive system does not perform as we expect. We will make trade-offs between instrumentality and intrusiveness, i.e., to balance between the ability to collect as much relevant information as we want and the amount of perturbation introduced.

- *Hardware support for formal methods*. Formal methods will be ever important in providing safety guarantees in cognitive computing. Hardware support is necessary to enable practical formal verification on large scale machine learning systems. The key is to realize that machine learning is a particular application domain that relies heavily on linear algebra, and thus it is possible to specialize the hardware support for formal verification to maximize efficiency.

4 Conclusion

Cognitive computing is redefining the way we perceive and interact with the world. As engineers who build systems, the responsibility rests upon us to design systems that enable safe and robust cognitive computing from the bottom-up, thereby pushing cognitive systems one step closer toward widespread usage. We hope that the perspectives we share in this article raise the awareness of cognitive computing safety in the architecture community and foster meaningful discussions that lead to new research in building safe cognitive computing.

References

[1] R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, "Fathom: Reference Workloads for Modern Deep Learning Methods," in *Proc. of IISWC*, 2016.

[2] BEA, "Final Report On the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro - Paris," July 2012.

[3] D. Beyer, *The Future of Machine Intelligence: Perspectives from Leading Practitioners*. O'Reilly, March 2016.

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proc. of DAC*, 2003.

[5] J. Chang, G. A. Reis, and D. I. August, "Automatic instruction-level software-only recovery," in *Prof. of DSN*, 2006.

[6] F. Cristian, "Understanding Fault-Tolerant Distributed Systems," *Communication of The ACM*, vol. 34, pp. 56–78, 1993.

[7] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[8] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000.

[9] EE Times, "ARM Raises Bar for Safety, Determinism," September, 2016. http://www.eetimes.com/document.asp?doc_id=1330483

[10] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: efficient inference engine on compressed deep neural network," in *Proc. of ISCA*, 2016.

[11] R. P. Pothukuchi, A. Ansari, P. Voulgaris, and J. Torrellas, "Using Multiple Input, Multiple Output Formal Control to Maximize Resource Efficiency in Architectures," in *Proc. of ISCA*, 2016.

[12] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernndez-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators," in *Prof. of ISCA*, 2016.

[13] D. J. Sorin, "Fault tolerant computer architecture," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–104, 2009.

[14] The Verge, "Tesla driver killed in crash with Autopilot active, NHTSA investigating," June 2016. http://www.theverge.com/2016/6/30/12072408/tesla-autopilot-car-crash-death-autonomous-model-s