Retrospective: A dynamically configurable coprocessor for convolutional neural networks

Srimat Chakradhar*, Murugan Sankaradas*, Venkata Jakkula, Srihari Cadambi†
*NEC Laboratories America, Inc.
†Google, Inc.

I. Serendipitous confluence of new and old

In 2008, parallel computing posed significant challenges due to the complexities of parallel programming and the bottlenecks associated with efficient parallel execution. Inspired by the remarkable scalability achieved by networking and storage systems in handling extensive packet traffic and persistent data respectively by leveraging best-effort service, we proposed a new and fundamentally different approach of best-effort computing [1].

Having observed that a broad spectrum of existing and emerging computing workloads were from applications that had an inherent forgiving nature [2], [5], we proposed best-effort computing. The new approach resulted in disproportionate gains in power, energy and latency, while improving performance.

While contemplating the concept of best-effort computing [2], we noticed the resurgence of convolutional neural networks, which generated approximate but acceptable outcomes for numerous recognition, mining, and synthesis tasks. The lead author of this retrospective had previously conducted research on neural networks for his doctoral dissertation over a decade ago, and the reemergence of neural networks proved both surprising and exciting. Recognizing the connection between best-effort computing and convolutional neural networks, in 2008 we embarked on developing a programmable and dynamically reconfigurable convolutional neural network capable of performing best-effort computing for various machine learning tasks that inherently allow for multiple acceptable answers. This combination of our thoughts on best-effort computing and the gradual evolution of convolutional neural networks (deep neural networks emerged much later) culminated in our 2010 ISCA work on dynamically reconfigurable convolutional neural networks.

II. Additional motivation for this work

Besides best-effort computing, the early 2000s saw broader trends like specialized chips becoming expensive requiring very high volumes to justify production. In addition, FPGAs were becoming more widespread and starting to handle complex workloads. FPGA chips offered significant amounts of low-latency on-chip memory, fast interconnection networks, and complex processing elements. Applications that utilized machine learning, and in particular convolutional neural networks (CNNs), were on the rise, especially in the image processing domain.

Around this time, NEC had already deployed products in image recognition and processing. This, along with interest from customers and best-effort computing, spurred forward-thinking research at NEC's US research facility, which resulted in this work. One of our objectives was to address the limitations of state-of-the-art processors at that time, particularly their inability to perform real-time complex pattern recognition in QVGA video. To tackle this challenge, we utilized an FPGA that could be reprogrammed to accommodate a specific convolutional neural network (CNN) architecture. This architecture incorporated essential components such as convolution units, dedicated sub-sampling logic, and nonlinear functions. To program this FPGA-based architecture, we employed very long instruction word (VLIW) instructions generated by our CNN compiler. The results were promising, and we achieved significant speedups over multi-core processors of that era.

III. Follow-up on our work

Since its initial publication, our work has garnered nearly 500 citations, illustrating its significant impact in the field. Several of these focus on mapping Convolutional Neural Networks (CNNs) to Field-Programmable Gate Arrays (FPGAs) or specialized custom hardware, and use compiler optimizations such as loop tiling to maximize FPGA performance [6], energy-efficient hardware designs [3], and designs that trade off accuracy to handle computationally and memory-intensive CNN layers. This is a testament to the continued importance of high performance for CNNs. Furthermore, the emphasis on addressing CNN bottlenecks and limitations through lower-level hardware enhancements, rather than relying solely on software advancements, still remains a promising direction in ongoing research.

IV. What we got right, and what we didn't

One of our observations was that because CNN topologies are so diverse in terms of the parallelism opportunities they offer, it made sense to tailor the hardware to a specific network. It was (and still is) impractical to build a chip for each network, but FPGAs offered us the ability to dynamically reconfigure the hardware on a per-network basis. Reconfigurability provided us flexibility and performance; we leveraged it to deal with an aspect of Moore's law where on-chip compute capability scaled faster than off-chip memory bandwidth. Fast-forwarding a decade and some years, some amount of reconfigurability is present in compute clusters today, but interestingly, FPGA-based accelerators are being seen as one choice to deal with the slowing of Moore's law. A somewhat different rationale, but a similar end result.

The other aspect that we got right was the use of a loosely-coupled coprocessor (that did not share memory with the host) for these types of workloads. The trend to use coprocessors, along with high off-chip memory bandwidth, have both been identified as important in the years following our publication, although our paper was likely not the only trigger for this trend.

We didn't really propose an alternative to the Von Neumann model of computation. Our architecture still leveraged memory to store intermediate data (some on-chip and some off-chip). More recent innovations in this space, such as the Tensor Processing Units (TPUs) [4] that were introduced around 2016, deployed systolic architectures to get around the memory bottleneck for neural network workloads. In effect, a different computation model from the traditional Von Neumann architecture has seen success.

We also didn't quite get the granularity and specificity of the hardware right: our solution was very tailored for CNNs in the sense that the hardware looked a lot like a CNN with convolvers, non-linearity, and so on. The success of hardware that is highly fine-tuned to this extent depends on how ubiquitous the application turns out to be. Machine learning is ubiquitous now, but it's more than just CNNs. So in retrospect, a little more generality could have taken us further.

V. Evolution of hardware accelerators

Since the publication of our work in 2010, hardware accelerators for convolutional neural networks (CNNs) have made notable advancements, leading to significantly improved performance. Nevertheless, two fundamental aspects have remained consistent throughout these developments. First, hardware accelerators leverage the inherent forgiving nature of numerous AI applications, allowing them to exploit efficiency gains. Second, the diversity in CNN models, encompassing variants such as transformers, underscores the necessity for practical hardware accelerators with dynamic reconfigurability. This ensures that the accelerator can adapt to different CNN models.

Graphics Processing Units (GPUs), originally designed for computer graphics, became popular for accelerating deep learning tasks, including CNNs. Since 2010, GPUs have continued to improve in terms of computational power, memory capacity, and memory bandwidth, making them a standard choice for accelerating machine learning tasks. Going further, Tensor Processing Units (TPUs) which are specifically designed for learning offer significant speedups over GPUs for those tasks. And recently, companies like Apple and Qualcomm have integrated CNN accelerators into SoCs (system-on-chip) to enable efficient on-device AI inference. This integration reduces power consumption, latency, and data transfer requirements, making CNN inference more practical for mobile and edge devices.

Field-Programmable Gate Arrays (FPGAs) have gained attention as flexible and programmable hardware accelerators for CNNs. They can be reconfigured to implement custom CNN architectures, and FPGAs have improved significantly in terms of resource utilization, memory bandwidth, and programmability since 2010.

Neural Processing Units (NPUs) have emerged as specialized hardware units designed specifically for neural network computations. They often include custom-designed processing elements optimized for CNN operations, such as convolution, pooling, and matrix multiplications.

Today, best-effort computing continues to enhance computational efficiency, and hardware accelerators continue to incorporate support for low-precision operations and quantization techniques. By using reduced precision for weights, activations and computations, these accelerators continue to exploit the inherent forgiving nature in recognition, synthesis and mining tasks. They routinely achieve higher throughput and energy efficiency without significant loss in accuracy, and dynamic reconfigurability is still as essential as we observed in 2010!

It is refreshing to see that although our work was done almost 15 years ago, there is continued enduring focus on leveraging forgiving nature and addressing dynamic reconfigurability in modern hardware accelerators for AI applications beyond CNNs.

Acknowledgment

We would like to thank Anand Raghunathan (currently a distinguished professor at Purdue U.) and Venkat Jakkula, our colleagues at NEC the time of writing the ISCA paper, for being an integral part of our broader pioneering effort on best-effort computing for machine learning applications. We would like to thank our summer intern in 2008 from University of Virginia, Jiayuan Meng, for patiently working with us to understand the forgiving nature of the few machine learning tasks (in image and video analytics) available at the time that had good open-source models or statistical algorithms!

References

[1] S. Chakradhar, J. Meng, and A. Raghunathan, "Best-effort parallel execution framework for recognition and mining applications," in *Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium*, 2009.

[2] S. Chakradhar and A. Raghunathan, "Best-effort computing: rethinking parallel hardware and software," in *Proceedings of the 2010 ACM/IEEE Design Automation Conference*, 2010, pp. 865–870.

[3] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[4] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[5] J. Meng, A. Raghunathan, and S. Chakradhar, "Exploiting the forgiving nature of applications for scalable parallel execution," in *Proceedings of the 2010 IEEE International Parallel and Distributed Processing Symposium*, 2010.

[6] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, 2015, pp. 161–170.