RETROSPECTIVE: Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks

Yu-Hsin Chen
EnCharge AI
yhchen@enchargeai.com

Joel S. Emer
MIT/NVIDIA
jsemer@mit.edu

Vivienne Sze
MIT
sze@mit.edu

I. Background

The origins of this 2016 ISCA paper date to early 2013, when two of the authors met during a faculty visit to MIT. There ensued a highly animated discussion about the consequences of the end of Moore's law and the opportunities in hardware architectures. Although there was significant agreement on the need and possibilities for hardware innovation, the two had slightly different perspectives on what was most important: with one focused on extreme efficiency through specialization (while providing sufficient application-level flexibility) and the other focused on maximizing flexibility to facilitate application innovation (while providing sufficiently improved efficiency). After Vivienne joined the faculty, they continued their discussions while searching for a concrete project.

An ideal project opportunity presented itself given the 2012 demonstrations of the capabilities of deep neural networks (DNNs) on various recognition tasks. We recognized that while DNNs provided high accuracy it came at a high computational cost, and therefore was a prime target for hardware support. This domain especially resonated with one of the authors who had been working on energy-efficient algorithms and hardware support for video compression (*i.e.*, compressing pixels) [1], and so DNN computation provided a natural evolution of that work into image recognition (*i.e.*, understanding pixels). This domain also resonated with the other author who was in the midst of developing a more general-purpose spatial architecture that could potentially target this new computation.

With the domain identified, we enticed a student to join the effort. This student had already fortuitously taken the relevant courses (computer architecture and machine learning), and he had already demonstrated in an earlier project his ability to tackle complex designs (he worked on the most complex block in video compression [2]) and also was able to think big picture and in a principled manner.

II. Development

The interesting, but challenging, properties of DNN workloads were that (1) they are so large that the data (weights and activations) could not be all kept on-chip; (2) the shape of the computation varied from layer to layer. While there were earlier efforts to design accelerators for DNNs, they often simplified the workload by reducing its size and fixing its shape. Unfortunately, these simplifications could affect the accuracy of the DNN model. In contrast, the authors decided that it would be interesting and of great value to directly tackle these challenges. They would not assume that the DNN model could fit on chip, meaning that whatever architecture they designed could support DNNs regardless of the scale. They would allow the shape of the DNN model to vary, so that their architecture would be flexible enough to not restrict the types of layers supported. This increased the likelihood that whatever design principles they developed would remain relevant as the DNN models continued to evolve.

It was already well-known that data movement is expensive in terms of both hardware performance and energy efficiency. And we quickly recognized that data movement could be defined in the form of a dataflow that specified how and where data is moved during the computation (*e.g.*, move activations or move weights). These principles guided the research to embark on an effort to formulate and theorize a framework to reason about the relationship between dataflow and hardware architectural properties.

At the same time, however, while trying to survey the existing work, we also found that the choice of dataflow was often embedded and implied as a part of the overall micro-architectural design. There was minimal modularity as each design was presented as uniquely optimized for its use cases. There also lacked a common terminology for describing dataflows in a precise yet succinct manner and further analysis of the existing work was very challenging.

These findings led the authors to develop a system that could precisely describe all possible dataflows. This was later recognized as the operation space [3], and each dataflow is a specific traversal through the space. We observed that there were certain common patterns of space/time traversal (*i.e.*, data movement) among the existing architectures. A salient feature across these patterns was the stationarity of data, such as the *weight-stationary* or *output-stationary* dataflows as we highlighted in this paper. We also found that it was very useful to reason about the architecture by separating out the dataflow as a distinct attribute of the architecture. By doing so, it was possible to easily describe a design first via its dataflow and then by other micro-architectural optimizations.

Given our new nomenclature, it became standard practice for us to analyze the the content of each new DNN accelerator paper in terms of these concepts. And invariably a description would arise of the form, "the paper presents a design based on a <blank>-stationary dataflow enhanced with these specific optimizations" (*e.g.*, a network or buffering strategy).

The vast space of possible dataflows, coupled with different hardware optimization techniques, have echoed the motivating

factors of the project to search for a balance between efficiency and flexibility. Our journey started with a spatial architecture (composed of an array of processing elements (PE)) that had already been researched by one of the authors [4] and a dataflow developed for the "edit distance" calculation [5] that used inter-PE communications, which we thought would be attractive to reduce data movement energy.

This experimentation guided the development of the Eyeriss architecture [6]. The Eyeriss dataflow, called *row stationary*, can be seen as the logical outcome of all of the lessons discussed above. First, rather than focusing on minimizing the data movement of a specific data type (weights, input activations, output activations), it focused on reducing the *overall data movement energy* which considers all data types.

Second, the varying shape and size of layers also dictated the need for *mapping* of the computation onto the hardware even when employing the same core dataflow. The existence of many mappings for a single DNN model allows an architecture to support multiple space/time traversals with varying speed and energy efficiency. Thus, the flexibility of an architecture is manifest through the overall space of valid mappings.

Last but not least, by recognizing the mapping requirements under the chosen dataflow, we could then maximize efficiency by stripping away hardware overheads. This resulted in a design with a very simple PE that has no cache or instructions as well as a very stylized buffer management scheme, similar to those used in video compression accelerators [7].

III. Impact

An immediate impact of the concepts developed for this paper is that they informed the architecture of the Eyeriss chip. The development of that chip validated that the dataflow was a key attribute of the design and contributed to the overall efficiency of the architecture. Eyeriss added flexibility by allowing for an expanded mapping space, which required only simple, efficient hardware (thus offering a good balance of efficiency and flexibility). This provided two benefits: First, by allowing for different mappings for a specific DNN model layer, we could find the most energy-efficient mapping. Second, expanding the mapping space allowed the chip to support a larger variety of DNN model layers. In specific, this allowed Eyeriss to support the shapes of all the layers of AlexNet and thus run what was at the time a large state-of-art DNN model with high efficiency. This included low DRAM traffic, even though the chip had limited on-chip storage. Furthermore, Eyeriss illustrated how a design can be understood as a dataflow enhanced with other micro-architectural features. The core Eyeriss dataflow was augmented with micro-architectural optimizations including exploiting input activation sparsity by gating activity and using compression to reduce data movement.

The architecture concepts in this paper also led to a number of derivative architectures, most directly Eyeriss v2 [8] and Tetris. And although the Eyeriss design has been superseded as the most optimal dataflow and architecture it is a baseline against which new designs are invariably compared.

Ideas in this paper have also impacted subsequent research, including formalizing the notions of data orchestration and specific data orchestration idioms [9] and the dataflow and mapping modeling in this paper was a direct antecedent of Timeloop [3] and many follow-on mapping efforts.

However, we feel the lasting impact of this paper is the method that we used to describe a DNN accelerator architecture. In specific, they can be characterized by their dataflow, and by separating out the dataflow as a discrete architectural facet of a design, thus making it easier to understand the designs of DNN (and other) accelerators. Also, this paper highlights that certain dataflows can be identified by the data that is kept stationary (*i.e.*, a *-stationary dataflow). This terminology has become deeply ingrained in our community (and beyond) and is core to the book we coauthored [10].

IV. Conclusion

This project taught us several important lessons in research. First, it is important to directly tackle the complex and difficult problem head-on as the resulting solutions can be long-lasting. Second, it is important to collaborate with people with complementary skill sets and sometimes distinctly different views (*e.g.*, emphasis on flexibility versus efficiency); it can result in a compromise solution that has a high impact as it satisfies multiple criteria. Finally, in this age of exponentially increasing number of publications, it is increasingly important to organize and formalize the design space in order to provide deeper long-lasting insights that advance understanding (scientific knowledge) and improve communication.

References

[1] V. Sze, M. Budagavi, and G. J. Sullivan, "High Efficiency Video Coding (HEVC): Algorithms and Architectures," in *Integrated Circuit and Systems*. Springer, 2014.

[2] Y.-H. Chen and V. Sze, "A deeply pipelined CABAC decoder for HEVC supporting level 6.2 high-tier applications," *Transactions on Circuits and Systems for Video Technology*, 2014.

[3] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *International Symposium on Performance Analysis of Systems and Software*, 2019.

[4] A. Parashar, M. Pellauer, M. Adler, B. Ahsan, N. Crago, D. Lustig, V. Pavlov, A. Zhai, M. Gambhir, A. Jaleel *et al.*, "Triggered instructions: A control paradigm for spatially-programmed architectures," *International Symposium on Computer Architecture*, 2013.

[5] J. J. Tithi, N. C. Crago, and J. S. Emer, "Exploiting spatial architectures for edit distance algorithms," in *International Symposium on Performance Analysis of Systems and Software*, 2014.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *Journal of Solid-State Circuits*, vol. 52, no. 1, 2017.

[7] V. Sze, D. F. Finchelstein, M. E. Sinangil, and A. P. Chandrakasan, "A 0.7-V 1.8-mW H. 264/AVC 720p video decoder," *Journal of Solid-State Circuits*, 2009.

[8] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.

[9] M. Pellauer, Y. S. Shao, J. Clemons, N. Crago, K. Hegde, R. Venkatesan, S. W. Keckler, C. W. Fletcher, and J. Emer, "Buffets: An efficient and composable storage idiom for explicit decoupled data orchestration," in *Architectural Support for Programming Languages and Operating Systems*, 2019.

[10] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks," *Synthesis Lectures on Computer Architecture*, 2020.