Thesis for The Degree of Licentiate In Computer Science and Engineering

Techniques to Improve Management of On-Chip Memories of Deep Learning Accelerators

Piyumal Ranawaka

Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden, 2025

Techniques to Improve Management of On-Chip Memories of Deep Learning Accelerators

Piyumal Ranawaka

ISSN 1652-876X

Department of Computer Science and Engineering
Division of Computer and Network Systems
Computer Architecture Research Group
Chalmers University of Technology | University of Gothenburg
SE-412 96 Göteborg,
Sweden
Phone: +46(0)31 772 1000

Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2025.

“Imagination is more important than knowledge.”
- Albert Einstein

Abstract

This thesis focuses on two open problems in the literature related to improving the management of on-chip memories of deep learning accelerators for improved resource utilization.

The first part of this thesis focuses on deep learning accelerators that suffer from poor on-chip memory utilization, affecting performance and energy efficiency. Techniques such as loop reordering and blocking are used to improve this, but existing frameworks can be inefficient due to either high algorithmic computational complexity for searching or, due to suboptimal choices due to compromised search space. This paper presents DNNOPT, a hardware/software framework that optimally selects loop orders and blocking factors through two proposed stratergies: *Early Exit* and *Strided Search*, to prune the search space and simple analytical models for data reuse. DNNOPT reduces the search space by over two orders of magnitude and enhances performance, energy efficiency, and time to solution by an average of 1.8×, 50%, and 226×, respectively, for CNN and Transformer workloads compared to current frameworks.

The second part of this thesis focuses on accelerators where multiple neural network inferences can run simultaneously that allow the simultaneous execution of multiple Deep Neural Network (DNN) workloads, improving performance by overlapping computations and memory access. For effective operation, sufficient on-chip memory is necessary to accommodate the total memory footprint of all workloads. Batching enhances weight reuse and lowers off-chip access costs by enabling DNN inferences of the same model to share weights. However, traditional batching, which sets the batch size statically across all layers, can cause stalls when on-chip memory is insufficient. This paper introduces BATCH-DNN, a dynamic method that adjusts the batch size of each layer based on available on-chip memory. It employs techniques such as *adaptive cascaded sub-batching* and *adaptive sub-batch merging.* Offline profiling determines the memory footprint, while runtime adjustments set the maximum batch size per layer. BATCH-DNN can boost accelerator compute fabric utilization by 40%, leading to throughput improvements of up to 27%, with an average enhancement of 6% for batched multi-DNN workloads.

Keywords

Deep Learning Accelerator On-Chip Memory, Loop Optimizations, Loop Reordering, Loop Blocking, Dynamic and Adaptive Batching

List of Publications

Appended publications

This thesis is based on the following publications. References to the papers are made using Roman numerals associated with the papers:

I. P. Ranawaka, M.W. Azhar, P. Stenstrom, *DNNOPT: A Framework for Efficiently Selecting On-chip Memory Loop Optimizations of DNN Accelerators*
In Proceedings of the *21st ACM International Conference on Computing Frontiers* (May 2024), 126-137.

II. P. Ranawaka, P. Stenstrom, *BATCH-DNN: Adaptive and Dynamic Batching for Multi-DNN Accelerators*
To appear in *31st European Conference on Parallel and Distributed Computing,* (August 25-29), 2025.

Other publications

The following publications were published during my PhD studies, However, they are not appended to this thesis, due to contents not related to the thesis.

[**a**] G. Baravdish, P. Ranawaka, *Semantic Segmentation of Weed and Crop with Partially Annotated Data for Automated Agriculture*
Proceedings of *2023 IEEE International Conference on Agrosystem Engineering, Technology & Applications (AGRETA)* (September 2023), 17-22.

Acknowledgment

First, I would like to express my sincere and profound gratitude to my supervisors, Prof. Ulf Assarsson and Prof. Per Stenstrom. They have been the best supervisors I could ever ask for—believing in me, motivating me, providing outstanding research training, and offering unwavering support. Ulf offered his excellent supervision and support during a brief period when I was in great need, for which I am deeply grateful. Per, on the other hand, supervised me for many years and without giving up on me. So I would like to say a few more words about Per since I know quite a lot about him. He is not only a patient listener filled with humility but also a mentor, an excellent teacher, an innovator, a role model, and a true leader in the research world of computer architecture. His expertise in the field is matched only by his personal qualities and integrity, making him someone from whom young researchers can learn so much. Professionally, I thank you for being a great leader, for trusting me and delegating responsibilities, for helping me grow professionally, and for being patient whether things went as expected or not. I aspire to be your faithful assistant, as you have been a wonderful boss. As the father in the research context, I will always remember you for teaching me to think in ways I had never used to before—such as abstract thinking, which involves focusing on concepts while ignoring details, and dialectical thinking, which entails holding opposing viewpoints simultaneously to reach a more nuanced understanding and many more approaches for complex reasoning. You are the wisest person I have ever encountered, always driven by reasoning. Your anecdotes have served as valuable life lessons. The lessons I learned from you about writing, not only scientific writing, coming from an expert writer, will be the most powerful weapon I possess. Your guidance in communication and presentation, particularly on how to pitch complex ideas and to make the points, will be invaluable skills to master. Finally, thank you for teaching this young kid, who was not accustomed to thinking before acting, that if you sit and think deeply, anything can be conquered. At first, I saw you as a philosopher, but now I understand that you are truly a remarkable intellectual and a scientist.

I would also like to express my deep appreciation and gratitude to my first two teachers, Dr. Mongkol Ekpanyapong and Prof. Adriano Tavares, who have significantly shaped who I am today with their elite skills and expertise. Their relentless support and guidance continue to benefit me. I still remember the first time I met them, coming from a country with an exam-oriented education system that offered free education but not freedom of education, and limited access to quality higher education and opportunities for growth, in my humble opinion. I was unfamiliar with managing complex technical projects, often struggling with even a few hundred lines of code and a few thousand transistors, and I didn't know how to use the state-of-the-art tools necessary to complete my work. I am incredibly grateful for their help in helping me master the craft and build my confidence and self-esteem through the best training imaginable. They fostered my problem-solving skills, creativity, patience, persistence, endurance, agility, and resilience, enabling me to manage a complex, intellectually challenging project. Having access to a former senior computer architect at Intel and a 'GOAT' in computer engineering who are incredibly kind and caring towards young students has been a significant turning point in my life.

In summary, meeting these outstanding teachers has profoundly fulfilled my desire to 'deeply' understand how computers work, a mystery that has intrigued me since childhood. I consider myself fortunate to have them as my mentors and pillars of strength. Finally, I want to thank them for offering me opportunities when all I had to offer was my willingness to learn, along with my limited knowledge and skills for the role. I would also like to thank Dr. Waqar for his friendly co-supervision and for putting up with my stubbornness.

I would like to express my heartfelt gratitude to my outstanding support team at Chalmers for their unwavering belief in me, for helping me succeed despite the numerous challenges and setbacks I faced, and for not giving up on me. I appreciate the excellent workplace they provided, particularly my understanding and caring manager, Arne, for his fatherly care and for picking me up when I am psychologically drained due to countless reasons beyond my control, along with my PhD team, including Ioannis, Agneta, Wolfgang, Nir, Clara, and Sana. I also want to thank all the esteemed faculty members in our department for their warm, inspiring, and welcoming presence, including Roger, Pedro, Ioannis, Miquel, Risat, Lars S., Johan, Tomas, Romaric, Ahmed, Niklas, and Richard. Furthermore, I am grateful to my colleagues in the department—Ahsen, Prajith, Albin, Waqar, Monica, Lars N., Andreas, Pirah, Fazeleh, Mehrzad, Qi, Konst, Mateo, Neethu, Bhavi, Madhavan, Sonia, Jing, Stavroula, Panos, Magnus, Mo, Francesco, Roman, and many others—for their warm companionship and for all the insightful and not so insightful discussions we've had.

I would like to express my gratitude to the teachers who supported me throughout my higher studies at SAITM Sri Lanka, AIT Thailand, University of Minho in Portugal. This includes Professors Matt, Cabral, Sisuru and Janaka; Doctors Tiago, Vitor, Krit, Thilini, Harsha, Chinthaka, and Tharanga, as well as Mr. Chumnarn, Mr. Marcelo, and Mr.Rukshan. I cannot forget the education I received from my home country, particularly from SAITM, where I created cherished memories and gained foundational lessons in engineering.

I also want to acknowledge my alma mater, St. Joseph's College in Colombo, and Nalanda College in Colombo, where I had remarkable teachers, including Mr.Christopher Gamage, Mrs.Shriya Perera, and Mr.Doric Jayakody. How could I ever forget beloved and excellent peers like the late Nisal Kotinkaduwa for inspiring and pushing hard a grade ten student to do a PhD one day. My schools instilled in me the values of 'In Scientia et Virtute' and 'Apadana Sobhini Panna.'

Further, a big acknowledgment to all the teachers, but hard to attribute to one, who instilled an attitude and a virtue of do it with all your ability pouring all your heart and soul into whatever you do or do nothing.

I would like to extend my gratitude to the University of Moratuwa. While it is not my alma mater, it sparked my passion for engineering during a time when my school education felt boring and uninspiring. I am thankful for the opportunity to serve as a junior faculty member there, and I appreciate the friendly and supportive staff members, including Prof.K.K.C.K Perera, Mr.Sudantha, Mr.Karunarathne, Dr.Lochandaka, Dr.Supunmali, Dr.Amalraj, Dr.Saminda, Mrs.Indika, Mr.Withanage, Dr. Shalinda, Dr.Chaman, Dr.Upeksha, Dr.Sagara, Dr.Ruvini, Dr.Sumudu, Prof.Karunananda, Dr.Leelanga, Dr.Subha, Prof. Thushari, Prof.Thanuja, and Dr.Champika. Their support has played a constructive role in advancing my career.

Last but not least, I would like to express my heartfelt gratitude to my dearest parents, grandparents, and sister for believing in me when no one else in the family did. Your encouragement has helped me become who I am today. I will always remember my parents for their endless love, commitment, and sacrifices. They instilled in me the importance of learning even before I could ride a bicycle or tie my shoelaces!

I am profoundly grateful to the Wallenberg AI, Autonomous Systems, and Software Program (WASP) for the opportunity to broaden my horizons, gain valuable knowledge, and build lasting friendships with fellow researchers. This work has been supported by WASP, which is funded by the Knut and Alice Wallenberg Foundation, and the PRIDE project, supported by the Swedish Foundation of Strategic Research under the CHI19- 0048 contract. The computations were made possible through resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), which is partially funded by the Swedish Research Council under grant agreement no. 2022-06725.

Finally, I express my profound gratitude to my beloved division, department and Chalmers, and to all taxpayers of Sweden and the EU for giving me the privilege and access to cutting-edge, world-class education at its highest academic quality.

Contents

Chapter 1

Introduction

1.1 Background

Deep Learning (DL) has become essential for many emerging applications because of its effectiveness in tackling complex tasks across various fields [1]–[4]. However, these applications are compute and data intensive, leading to poor performance and high power consumption when executed on general-purpose hardware. In addition, the end of Moore's law [5]–[7] implies that specialization is a promising avenue for future computer architecture where the key idea is to offload such demanding workloads to domain-specific hardware accelerators. Such accelerators eliminate performance and power overheads in general-purpose processing by using specialized hardware optimized for a specific class of applications. As a result, DL inference accelerators have become an integral part of computing systems from edge to cloud. On-chip memory in such accelerators is a key resource that improves performance and lowers energy consumption by reducing off-chip memory accesses by reusing data.

First, focusing on systems in which resources are constrained and, in particular, the amount of on-chip memory, such as in edge systems, the amount of on-chip memory resources that could be allocated to an inference is limited. As a result of having on-chip memory constrained, often, the entire footprint of a layer could not be accommodated in the on-chip memory. Limited on-chip memory results in frequent off-chip memory accesses that degrade performance and increase power consumption. Loop optimizations such as loop reordering and loop blocking have been extensively explored in prior art [8]–[29]. It is used as a means to improve the reuse of data in the on-chip memory and to cut down off-chip memory accesses. It does so by improving the locality of data access while adhering to the on-chip memory budget.

Second, there are systems with less strict resource constraints, such as in cloud systems, but achieving cost-effectiveness and higher throughput for the given resource investment is essential. This requires efficiently utilizing the compute fabric, off-chip memory bandwidth, and on-chip memory. Prior art [30]–[51] has attempted to enhance the cost-effectiveness by executing multiple DNN workloads simultaneously on a single accelerator. Some work uses double buffering to improve resource utilization by overlapping memory access with computation [34], [41], [43]–[46]. However, only a few studies focus on techniques to enhance resource utilization due to the mismatch between memory access and computation [30], [31], [33], [40]. Furthermore, less attention has been paid to constraints due to on-chip memory capacity. In cloud inference, batching means loading a layer's weights once to compute N consecutive inferences (a batch) before progressing to the next layer. This reduces off-chip accesses for weights by a factor of the batch size N but puts pressure on memory since the footprint of feature maps increases by a factor of N, where feature map data of N inferences must be accommodated in the on-chip memory simultaneously. Having to accommodate multiple such workloads in on-chip memory further exacerbates the pressure on the on-chip memory.

1.2 Problem Statement

The loop order and blocking factors are determined by the compiler with a computationally demanding process involving an iterative search also called *mapping* [8]–[20], while some others [11], [21]–[24] rely on performance models with high training cost or necessitate massive datasets and large-scale simulations to train them. However, there is often a tradeoff between computational complexity and optimality, where achieving optimality comes at the expense of computational complexity of search or vice versa. There have been some strides towards exploiting this tradeoff [9], [17]–[20]. However, the computational complexity of mapping an entire workload that consists of a large number of layers still remains high. Therefore, Paper I addresses the following question:

How to achieve optimality and low computational complexity simultaneously in the iterative search for loop orders and blocking factors in DNN accelerators?

Due to the dynamic nature of the cloud multi-DNN accelerators, using a fixed static batch size, *across all layers* of a workload, often leads to stalls and under-utilization when on-chip memory space is exhausted. This hampers performance measured as throughput. Conversely, smaller batch sizes reduce such stalls but can lead to poor throughput due to poor weight reuse. Therefore, Paper II addresses the following question:

How to mitigate the performance degradation caused by stalls due to exhaustion of on-chip memory when multiple statically determined batches share on-chip memory?

1.3 Thesis Contributions

This thesis is based on two papers. The main contributions of *Paper I*, which addresses the first question, are:

- We propose DNN-OPT, a hardware/software framework designed for output stationary systolic array-based accelerators. DNN-OPT selects the loop order and blocking factor on a layer-by-layer basis, either in isolation or in combination.

- Our results demonstrate that DNN-OPT can optimally minimize the number of off-chip memory accesses while significantly reducing computational complexity. This is achieved through two proposed techniques—*Early Exit* and *Strided Search* as well as the use of simple, computationally inexpensive analytical cost models tailored for systolic array-based accelerators.

The main contributions of *Paper II*, which addresses the second question, are:

- We propose two intertwined techniques, *adaptive cascaded sub-batching* and *adaptive sub-batch merging*, aimed at enhancing the utilization of the compute fabric and on-chip memory in multi-DNN accelerators.

- We evaluate the two proposed techniques. BATCH-DNN can improve the utilization of the accelerator compute fabric by 60%, increasing throughput by up to 27% and an average of 6% for batched multi-DNN workloads.

The remainder of the thesis is organized as follows: Chapter 2 provides a summary of each paper, while Chapter 3 concludes the thesis and presents potential future research directions.

Chapter 2

Summary of Papers

2.1 Summary of Paper I

Loop reordering and blocking are well-researched optimizations aimed at maximizing data reuse in deep neural network (DNN) accelerators. However, existing frameworks have limitations to efficiently select the optimal loop order and blocking factors for these accelerators. Specifically, prior approaches face one or more of the following limitations: (1) they require exhaustive searches through an excessively large search space, (2) they do not guarantee the selection of optimal optimizations if only a subset of the entire search space is evaluated or (3) they involve computationally intensive evaluations for each point in the search space for a given loop order and a set of blocking factors.

First, some studies, such as those referenced in [8]–[16], have attempted to exhaustively identify the optimization that minimizes execution time, resulting in a computational complexity of $O(M \times P)$[17]. In this case, M represents the number of operations needed to evaluate a single point, while P is the total number of points in the search space. Although there have been efforts to prune the search space (i.e, reduce P) [9], [17]–[20] it remains prohibitively large, particularly since it must account for all valid sizes of blocking factors and their combinations. Other work has employed feedback-driven approaches, such as black-box auto-tuning, beam search, Monte Carlo methods, or iterative sampling using machine learning algorithms [11], [21]–[24]. Unfortunately, these methods often incur high training costs or require extensive datasets and large-scale simulations to develop performance modelsas pointed out by [20].

Second, Timeloop [8], [25] utilizes random search methods, while some studies [26], [27] have applied genetic algorithms. Sometimes, these approaches lead to non-optimal results. For instance, Timeloop's random search can select points that result in performance five times worse than the optimal point available in the search space as pointed out by [20].

Third, focusing on reducing M, evaluating a single point in the search space within Timeloop involves examining delta in two contiguous tiles for data reuse. Similarly, Rahaman et al. [13] analyze every location in the on-chip memory using simulations. This leads to a high computational cost when assessing each individual point in the search space.

In response to these challenges, DNNOPT introduces a hardware/software framework designed to optimally and efficiently select loop order and blocking factors. One key challenge is how to reduce the search space (i.e., P) without sacrificing global optimality. To address this, we employ two strategies: (1) *Early Exit* from the search process if increasing the blocking factor does not result in fewer off-chip accesses. To elaborate, the cost models indicate that the number of off-chip accesses varies with the blocking factor sizes as a monotonic function. If increasing a blocking factor does not reduce off-chip accesses, it is unproductive to keep increasing that blocking factor. Thus, searching along this path can be terminated early. We show experimentally that the Early Exit strategy can significantly reduce search time. (2) *Strided Search* involves selectively traversing the search space to examine only valid block sizes relevant to the systolic array-based architecture. To elaborate on each layer, we search by iterating through block sizes and combinations. In an output stationary systolic array architecture, block sizes must be integer multiples of PE array size. To optimize the search, we propose *striding*, by which we mean that we evaluate only such block sizes that are integer multiples of the PE array size.

The second challenge involves reducing the computational complexity associated with evaluating a single point in the search space (i.e., M). To address this DNNOPT leverages precise and computationally inexpensive analytical cost models based on reuse distance and inter-block reuse, inspired by the work of [28], [29].DNNOPT also allows a quantitative comparison of the benefits of chosen optimizations for loop reordering and blocking, whether considered in isolation or in combination. Additionally, it selects loop order and blocking factors on a layer-by-layer basis.

We extend the SCALE-Sim simulator to model our optimizations. The baseline system suitable for resource-constrained devices is assumed. We choose a wide variety of DNN benchmarks selected from MLPerf [52], Deep Bench benchmarks [53], and benchmarks used in previous work [54], [55].

We present an evaluation of the framework's effectiveness, demonstrating that our proposed methods can prune the search space by over 99%, or two orders of magnitude, for both CNN and transformer workloads. Furthermore, our framework achieves an average of 1.8× improvement in performance, 50% improvement in energy efficiency, and 249× improvement in time to solution compared to Timeloop's random search [8], as well as a 226× estimated reduction in time to solution compared to CoSA [20] for CNN and transformer benchmarks.

2.2 Summary of Paper II

This paper addresses the problem of throughput degradation in cloud systems that serve multiple statically batched DNNs due to the exhaustion of on-chip memory. In this regard, we present dynamic techniques designed to adapt the batch size to suit the available amount of on-chip memory on a layer-by-layer basis. We propose two methods: *adaptive cascaded sub-batch splitting* and *adaptive sub-batch merging.*

First, common to both techniques, we statically profile the workload for a single batch to determine the memory footprint required for each inference. Using this static information, the method can identify the maximum possible batch size for each new layer at runtime.

In cases where the batch size used in the previous layer exceeds the available amount of on-chip memory at any given layer, the batch is divided into two sub-batches, thereby reducing memory consumption. We refer to this as *Sub-batch splitting.* Furthermore, when the on-chip memory insufficiency is resolved, sub-batch merging combines the currently executing sub-batch with the most recently paused sub-batch, ensuring that layer dependencies are maintained. The paused sub-batches are resumed in the order in which they were paused to respect these dependencies. This is referred to as *Sub-batch Merging*

The two main challenges in deploying these techniques involve maintaining execution order and managing on-chip memory. This is particularly important because sub-batch splitting and merging can occur dynamically based on memory availability at runtime. First, all preceding dependent layers must complete their computations to produce Output Feature Maps (OFMAPs), which will be used as Input Feature Maps (IFMAPs) for the current layer. Second, the weights for the current layer must be loaded into on-chip memory before computations can begin. Once all dependent layers have finished processing, the weights and IFMAPs can be freed up. However, managing on-chip memory is challenging since batches may only complete partially due to sub-batching taking place at arbitrary layers.

The proposed sub-batching techniques exhibit a form of recursion, where a given sub-batch can be further divided into two sub-batches. This creates a binary tree structure of sub-batches, complicating the tracking of their splitting and merging. The last split sub-batch needs to be merged first. To address this, we propose using a stack to keep track of the paused sub-batches. Thus, sub-batches are resumed in Last-In-First-Out (LIFO) order. Each inference workload has a stack.

The BATCH-DNN scheduling algorithm divides batches into sub-batches when there is insufficient on-chip memory at runtime. It uses a stack to document the paused work. Once the memory issue is resolved, the sub-batches are resumed and combined with the current batch. This approach allows BATCH-DNN to dynamically adjust its batch size to the availability of on-chip memory during execution, thereby mitigating stalls.

We utilize BATCH-DNN_Sim, a cycle-accurate simulator [56], to profile Deep Neural Network (DNN) workloads. This simulator establishes their memory footprints, memory access patterns, and computation cycle counts based on an Output Stationary (OS) systolic array model. BATCH-DNN_Sim takes as inputs micro-architectural parameters, along with the DNN topology and populates the scheduling table.

This simulator further models a multi-DNN accelerator [56] by extending the AI-MT scheduler [33], generating cycle counts for computations, memory accesses, and any stalls that occur. The AI-MT architecture model serves as our baseline; however, we have increased the on-chip memory to an amount equivalent to that used in STfusion [31] to make our baseline more realistic. Additionally, we assume the use of High Bandwidth Memory (HBM) for off-chip memory technology.

For our evaluation, conducting an exhaustive analysis of all combinations of four benchmarks would require considering $10 \times 10 \times 10 \times 10 = 10^4$ workload combinations, which is far too large to be tractable. Instead, we have chosen to evaluate 450 mixes from this exhaustive set randomly. Actually, 370 mixes achieve 95% statistical confidence with 5% error on a population of 10,000.

We evaluate the two proposed techniques. BATCH-DNN can improve the utilization of the accelerator compute fabric by 60%, resulting in a throughput increase of up to 27% and an average of 6% for batched multi-DNN workloads.

Chapter 3

Discussion and Future Work

Paper I introduces DNNOPT, a hardware/software framework designed for output stationary systolic array-based accelerators. DNNOPT optimizes the loop order and blocking factors layer-by-layer to enhance the performance and energy efficiency of deep neural network (DNN) accelerators by minimizing the number of off-chip memory accesses. Unlike prior work, DNNOPT makes optimal selections with significantly reduced computational complexity. It achieves this through three strategies: Early Exit, Strided Search, and the use of simple, computationally inexpensive analytical cost models based on global reuse distance and inter-block reuse.

First, a major drawback of DNNOPT is that it is tailored to a specific data flow, namely output stationary. Consequently, analytical models must be manually derived for each new data flow. In contrast, other tools, such as Timeloop and CoSA, are more versatile and applicable to all DNN accelerators. This limitation illustrates the typical trade-off between efficiency and generalization. Second, the DNNOPT architecture is restricted to two levels of tiling. It also relies on hardware tags in the memory allocation table to ensure that the memory blocking tiles remain transparent to the compute tiles. However, the hardware investment required for DNNOPT may still be justified for a memory space-constrained DNN accelerator that requires only a few tags. Third, DNNOPT is limited to six dimensions, excluding the batch dimension, which is less relevant in a memory-constrained context where it cannot accommodate the full footprint of a batch of tiles. Fourth, DNN-OPT mapping happens layer by layer, considering only a single layer at a time (i.e, considering intra-layer reuse only).

Future work of DNN-OPT includes incorporating both inter and intra-layer reuse in the mapping, given that layers are processed one after another with an on-chip memory with persistent data across layers.

Paper II presents BATCH-DNN, a framework for adaptive and dynamic batching in multi-DNN accelerators. BATCH-DNN helps reduce performance losses due to memory exhaustion by employing adaptive cascaded sub-batching and sub-batch merging techniques.

The main drawbacks of BATCH-DNN are as follows. First, the system can stall if it cannot accommodate at least one unit batch. Second, the baseline scheduler occasionally encounters deadlocks because memory access occurs out of order. This issue could be resolved by enforcing in-order memory access or handling it as an exception. Third, batch sizes that exceed the total on-chip memory capacity—meaning any given layer of a DNN exceeds the available memory—are blocked from processing, limiting the maximum attainable batch size.

The future work of BATCH-DNN could be extended to support arbitrarily larger batches with an approach toward squeezing a larger number of inferences in a given memory space with a layer dependency centric scheduler that is more synergistic with BATCH-DNN, which can reach the theoretical upper bound of batch size with fewer stall cycles. Further, an approach called batch slicing allows breaking larger batches into smaller ones before feeding them to the accelerator in a repetitive fashion; this lifts the constraint on the maximum servable batch size.

Further, it would be interesting to explore if both DNNOPT and BATCH-DNN could be combined to achieve a completely stall-free execution of batched DNNs with real-time dynamic mapping, optimal for a given memory allocation at runtime.

Bibliography

[1] Y. Tian, K. Pei, S. Jana and B. Ray, “Deeptest: Automated testing of deep-neural-network-driven autonomous cars,” in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 303–314 (cit. on p. 1).

[2] S. Lee, S. W. Oh, D. Won and S. J. Kim, “Copy-and-paste networks for deep video inpainting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4413–4421 (cit. on p. 1).

[3] V. Ramanishka, Y.-T. Chen, T. Misu and K. Saenko, “Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7699–7707 (cit. on p. 1).

[4] C.-J. Wu, D. Brooks, K. Chen *et al.*, “Machine learning at facebook: Understanding inference at the edge,” in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2019, pp. 331–344 (cit. on p. 1).

[5] N. Hardavellas, M. Ferdman, B. Falsafi and A. Ailamaki, “Toward dark silicon in servers,” *IEEE Micro*, vol. 31, no. 4, pp. 6–15, 2011 (cit. on p. 1).

[6] G. Venkatesh, J. Sampson, N. Goulding *et al.*, “Conservation cores: Reducing the energy of mature computations,” *ACM Sigplan Notices*, vol. 45, no. 3, pp. 205–218, 2010 (cit. on p. 1).

[7] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam and D. Burger, “Dark silicon and the end of multicore scaling,” in *2011 38th Annual International Symposium On Computer Architecture (ISCA)*, IEEE, 2011, pp. 365–376 (cit. on p. 1).

[8] A. Parashar, P. Raina, Y. S. Shao *et al.*, “Timeloop: A systematic approach to dnn accelerator evaluation,” in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2019, pp. 304–315 (cit. on pp. 1, 2, 5, 6).

[9] L. Mei, P. Houshmand, V. Jain, S. Giraldo and M. Verhelst, “Zigzag: Enlarging joint architecture-mapping design space exploration for dnn accelerators,” *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1160–1174, 2021 (cit. on pp. 1, 2, 5).

[10] S. Zhang *et al.*, “Pame: Precision-aware multi-exit dnn serving for reducing latencies of batched inferences,” in *ICS*, 2022, pp. 1–12 (cit. on pp. 1, 2, 5).

[11] X. Yang, M. Gao, Q. Liu *et al.*, “Interstellar: Using halide’s scheduling language to analyze dnn accelerators,” in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 369–383 (cit. on pp. 1, 2, 5).

[12] R. V. W. Putra, M. A. Hanif and M. Shafique, “Romanet: Fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 702–715, 2021 (cit. on pp. 1, 2, 5).

[13] A. Rahman, S. Oh, J. Lee and K. Choi, “Design space exploration of fpga accelerators for convolutional neural networks,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, IEEE, 2017, pp. 1147–1152 (cit. on pp. 1, 2, 5).

[14] K. Yang, S. Wang, J. Zhou and T. Yoshimura, “Energy-efficient scheduling method with cross-loop model for resource-limited cnn accelerator designs,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2017, pp. 1–4 (cit. on pp. 1, 2, 5).

[15] Y. Ma, Y. Cao, S. Vrudhula and J.-s. Seo, “Optimizing loop operation and dataflow in fpga acceleration of deep convolutional neural networks,” in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 45–54 (cit. on pp. 1, 2, 5).

[16] R. Venkatesan, Y. S. Shao, M. Wang *et al.*, “Magnet: A modular accelerator generator for neural networks,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2019, pp. 1–8 (cit. on pp. 1, 2, 5).

[17] P. Chatarasi, H. Kwon, A. Parashar, M. Pellauer, T. Krishna and V. Sarkar, “Marvel: A data-centric approach for mapping deep learning operators on spatial accelerators,” *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 19, no. 1, pp. 1–26, 2021 (cit. on pp. 1, 2, 5).

[18] A. Symons, L. Mei and M. Verhelst, “Loma: Fast auto-scheduling on dnn accelerators through loop-order-based memory allocation,” in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, IEEE, 2021, pp. 1–4 (cit. on pp. 1, 2, 5).

[19] S. Dave, Y. Kim, S. Avancha, K. Lee and A. Shrivastava, “Dmazerunner: Executing perfectly nested loops on dataflow accelerators,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–27, 2019 (cit. on pp. 1, 2, 5).

[20] Q. Huang, M. Kang, G. Dinh *et al.*, "Cosa: Scheduling by constrained optimization for spatial accelerators," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2021, pp. 554–566 (cit. on pp. 1, 2, 5, 6).

[21] J. Ragan-Kelley, A. Adams, D. Sharlet *et al.*, "Halide: Decoupling algorithms from schedules for high-performance image processing," *Communications of the ACM*, vol. 61, no. 1, pp. 106–115, 2017 (cit. on pp. 1, 2, 5).

[22] T. Chen, T. Moreau, Z. Jiang *et al.*, "{Tvm}: An automated {end-to-end} optimizing compiler for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 578–594 (cit. on pp. 1, 2, 5).

[23] H. Kwon *et al.*, "Heterogeneous dataflow accelerators for multi-dnn workloads," in *HPCA*, IEEE, 2021 (cit. on pp. 1, 2, 5).

[24] S. Kang *et al.*, "Ganpu: An energy-efficient multi-dnn training processor for gans with speculative dual-sparsity exploitation," *IEEE Journal of Solid-State Circuits*, 2021 (cit. on pp. 1, 2, 5).

[25] M. Horeni, P. Taheri, P.-A. Tsai, A. Parashar, J. Emer and S. Joshi, "Ruby: Improving hardware efficiency for tensor algebra accelerators through imperfect factorization," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2022, pp. 254–266 (cit. on pp. 1, 5).

[26] Y. Yu, Y. Li, S. Che, N. K. Jha and W. Zhang, "Software-defined design space exploration for an efficient dnn accelerator architecture," *IEEE Transactions on Computers*, vol. 70, no. 1, pp. 45–56, 2020 (cit. on pp. 1, 5).

[27] S.-C. Kao and T. Krishna, "Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, IEEE, 2020, pp. 1–9 (cit. on pp. 1, 5).

[28] A. Stoutchinin, F. Conti and L. Benini, "Optimally scheduling cnn convolutions for efficient memory access," *arXiv preprint arXiv:1902.01492*, 2019 (cit. on pp. 1, 6).

[29] M. Peemen, B. Mesman and H. Corporaal, "Inter-tile reuse optimization applied to bandwidth constrained embedded accelerators," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2015, pp. 169–174 (cit. on pp. 1, 6).

[30] Y. H. Oh *et al.*, "Layerweaver: Maximizing resource utilization of neural processing units via layer-wise scheduling," in *HPCA*, IEEE, 2021 (cit. on pp. 1, 2).

[31] E. Baek *et al.*, "Stfusion:fast and flexible multi-nn execution using spatio-temporal block fusion and memory management," *IEEE Trans.Comput.*, 2022 (cit. on pp. 1, 2, 8).

[32] S. Ghodrati *et al.*, “Planaria: Dynamic architecture fission for spatial multi-tenant acceleration of deep neural networks,” in *MICRO*, IEEE, 2020 (cit. on p. 1).

[33] E. Baek *et al.*, “A multi-neural network acceleration architecture,” in *ISCA*, IEEE, 2020, pp. 940–953 (cit. on pp. 1, 2, 8).

[34] Y. Choi *et al.*, “Prema: A predictive multi-task scheduling algorithm for preemptible neural processing units,” in *HPCA*, IEEE, 2020, pp. 220–233 (cit. on pp. 1, 2).

[35] J. Lee *et al.*, “Dataflow mirroring: Architectural support for highly efficient fine-grained spatial multitasking on systolic-array npus,” in *DAC*, IEEE, 2021, pp. 247–252 (cit. on p. 1).

[36] M. Reshadi *et al.*, “Dynamic resource partitioning for multi-tenant systolic array based dnn accelerator,” 2023 (cit. on p. 1).

[37] G. Shomron *et al.*, “Smt-sa: Simultaneous multithreading in systolic arrays,” *IEEE CAL*, 2019 (cit. on p. 1).

[38] G. Shomron *et al.*, “Non-blocking simultaneous multithreading: Embracing the resiliency of deep neural networks,” in *MICRO*, IEEE, 2020, pp. 256–269 (cit. on p. 1).

[39] C. Wang *et al.*, “Cd-msa: Cooperative and deadline-aware scheduling for efficient multi-tenancy on dnn accelerators,” *IEEE TPDS*, 2023 (cit. on p. 1).

[40] Y. H. Oh *et al.*, “Layerweaver+: A qos-aware layer-wise dnn scheduler for multi-tenant neural processing units,” *IEICE TRANS. on Information and Systems*, 2022 (cit. on pp. 1, 2).

[41] J. Choi *et al.*, “Enabling fine-grained spatial multitasking on systolic-array npus using dataflow mirroring,” *IEEE Transactions on Computers*, 2023 (cit. on pp. 1, 2).

[42] Y. Choi *et al.*, “Lazy batching: An sla-aware batching system for cloud machine learning inference,” in *HPCA*, IEEE, 2021 (cit. on p. 1).

[43] C. Li *et al.*, “Memory-computing decoupling: A dnn multitasking accelerator with adaptive data arrangement,” *IEEE TCADs*, 2022 (cit. on pp. 1, 2).

[44] J. Yang *et al.*, “Versa-dnn: A versatile architecture enabling high-performance and energy-efficient multi-dnn acceleration,” *IEEE TPDS*, 2023 (cit. on pp. 1, 2).

[45] L. Yin *et al.*, “Polyform: A versatile architecture for multi-dnn execution via spatial and temporal acceleration,” in *ICCD*, IEEE, 2023 (cit. on pp. 1, 2).

[46] J. Yang *et al.*, “Venus: A versatile deep neural network accel-erator architecture design for multiple applications,” in *DAC*, 2023 (cit. on pp. 1, 2).

[47] Y. Li *et al.*, “A silicon photonic multi-dnn accelerator,” in *PACT*, IEEE, 2023, pp. 238–249 (cit. on p. 1).

[48] Y. Li *et al.*, “A high-performance and energy-efficient photonic architecture for multi-dnn acceleration,” *IEEE TPDS*, 2023 (cit. on p. 1).

[49] M. Drumond *et al.*, “Equinox: Training (for free) on a custom inference accelerator,” in *MICRO-54*, 2021, pp. 421–433 (cit. on p. 1).

[50] S. Kim *et al.*, “Moca: Memory-centric, adaptive execution for multi-tenant deep neural networks,” in *(HPCA)*, IEEE, 2023, pp. 828–841 (cit. on p. 1).

[51] J. Shin *et al.*, “Algorithm/architecture co-design for energy-efficient acceleration of multi-task dnn,” in *ACM/IEEE DAC*, 2022, pp. 253–258 (cit. on p. 1).

[52] V. J. Reddi, C. Cheng, D. Kanter *et al.*, “Mlperf inference benchmark,” in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2020, pp. 446–459 (cit. on p. 6).

[53] *Deepbench*, GitHub, Jan. 2023. [Online]. Available: `https://github.com/baidu-research/DeepBench#inference-benchmark` (visited on 05/01/2023) (cit. on p. 6).

[54] 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 (cit. on p. 6).

[55] Y.-H. Chen, T.-J. Yang, J. Emer and V. Sze, “Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019 (cit. on p. 6).

[56] “Batchdnn_sim,” 2025. [Online]. Available: `drive.google.com/drive/folders/11EDgM6dKx1Zpl1\_g31EapBzpHs-sLbzj?usp=sharing` (cit. on pp. 7, 8).