Research Statement

Sai Qian Zhang
www.saiqianzhang.com

1 Overview and Research Vision

In the recent years, we have seen a proliferation of sophisticated Deep Neural Network (DNN) architectures that have achieved state-of-the-art performances across a variety of domains [7, 27, 31]. However, the algorithmic superiority of DNNs comes at extremely high computation and memory costs, which pose significant challenges to the hardware platforms executing them. In recent past, GPUs represented the cutting edge of DNN hardware. However, the ever-increasing complexity of DNNs has led to a quest for the next wave of improvement in DNN processing efficiency, which further expedites the development for the customized AI hardware accelerator (e.g., Google's Tensor Processor Unit [10]).

Building a highly efficient AI accelerating system is extremely difficult, as traditional DNN architectures are designed solely for the purpose of accuracy maximization without considering their hardware implementation cost. In contrast, we should adopt a methodology of hardware and DNN algorithm co-design, given the fact that the DNN architectures and the hardware platform executing them are tightly coupled and tangled, a slight modification on one of them may significantly impact the other. With this methodology in mind, my PhD research has been focused on the boundary between deep learning and hardware system design. Specifically, I developed efficient DNN architectures and algorithms that achieve superior hardware performance than prior work while maintaining near a state-of-the-art accuracy performance. Additionally, I designed the hardware circuits that take advantage of the architectural properties of the proposed DNNs to enable optimal energy efficiency and latency on the hardware platform (e.g., FPGA). Unlike most of the previous literature that proposes efficient DNN solutions without (or with limited) hardware evaluation, my work [36, 15, 14, 25, 35, 38, 37] includes a full-stack approach with a detailed hardware design and evaluation. I believe developing such co-design techniques is of paramount importance in the sense that it can serve as a bridge between the research communities of AI and the hardware system. **The overarching goal of my research is to develop efficient software algorithms and supporting hardware architectures by exploring the novel computing paradigms for the compute-intensive applications (e.g., machine learning)**.

A complex scientific research problem often involves multiple fields of study. Cross-disciplinary study enables the researchers to broaden their research horizons, and it encourages the researchers to confront questions that traditional disciplines do not ask while opening up new areas of research. Recently, deep Reinforcement Learning (RL) has been widely adopted to various real-world problems (e.g., traffic management, robot control). Recent works [26, 22, 12] have shown that RL can be applied to facilitate the hardware system design, which used to take significant amount of efforts from the experienced engineers to design, verify, and manufacture. This motivated me to explore the field of RL and its application on hardware circuit design. A summary of my PhD works is shown in Figure 1[1]. Overall, I have worked on three areas during my PhD study: 1. algorithm and hardware co-design for efficient DNN execution (red blocks in Figure 1). 2. reinforcement learning and its application (blue blocks in Figure 1). 3. the intersection between the above two fields (magenta block in Figure 1).

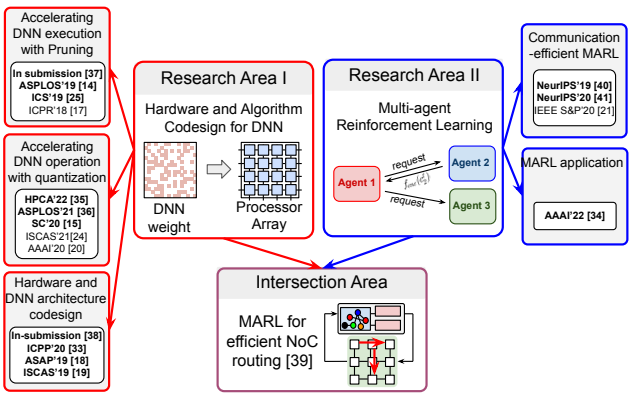

Figure 1: A summary of my PhD works. My first-author (or co-first author) publications are highlighted in bold.

The broader impacts of my work reach both academia and industry. Nine of my first-authored (or first co-authored) papers have been published at top tier architecture, machine learning or system venues (ASPLOS$\times$2, NeurIPS$\times$2, HPCA, SC, AAAI, ICPP, ICS). One of my first-authored paper has won the best paper award at IEEE International Conference in Communication (ICC). At Meta reality lab, I lead our ongoing collaborations with other industrial partners including TSMC and SK Hynix. During my doctoral study at Harvard, I have done multiple internships at Microsoft Research, Mediatek and Intel Labs. My works during my internships have resulted in two academic publications and one U.S. patent.

The rest of the statement will provide a detail of my research contributions in algorithm and hardware co-design (Section 2) and multi-agent reinforcement learning (Section 3). Finally, I will lay out plans for my future research (Section 4).

[1]For [14, 18, 19, 16], the first three authors contribute equally to the paper, proof with signatures from the authors can be provided upon request.

2 Algorithm and Hardware Co-design for Efficient DNN Deployment

To co-design DNN with hardware for a better efficiency, I have worked on developing innovative DNN pruning (removing superfluous parameters from DNNs) [37, 14, 25, 17] and quantization algorithms [36, 35, 15, 20, 24] (converting the high-precision DNN parameters into low-precision parameters). A common theme in these works is the usage of group-based methods that divide the matrix-matrix multiplication between a weight and data matrix into partial dot products. By limiting the computational cost on the dot products, it enables a tighter processing bound on the processing elements and further saves the overall processing latency and energy consumption. In addition to quantization and pruning, I also design novel DNN architectures and their supporting hardware for efficient DNN inference and training [33, 38, 18, 19].

2.1 Fine-grained Structured Pruning for Efficient Systolic Array Implementation [14, 25, 37]

For certain applications, it has been shown that DNN weight matrices can be sparse, e.g., with up to 90% of elements in the matrix being zero, while not significantly impacting the classification accuracy of the network [6]. Theoretically, such sparse matrices should lead to a significant reduction in inference computation, with a corresponding improvement in energy efficiency, as multiplication with zero weights can be skipped. However, in practice, the irregular positions of the nonzero weights make efficient hardware implementations difficult as they typically rely on a regular layout of processing elements. Based on this, I developed *column combining* [14], which packs sparse filter matrices into a denser format for efficient deployment in a novel systolic architecture. Column combining modifies conventional weight pruning by adding constraints across neighboring weight matrix columns. These constraints allow for a dense systolic layout to support sparse matrix multiplication and leads to significant improvement in energy efficiency (e.g., up to $8\times$) over conventional systolic arrays for sparse DNNs (Figure 2). Recently, column combining has been implemented by Nvidia in their A100 Tensor Core GPU [1], which supports a fine-grained structured sparsity by packing every four consecutive weight matrix columns into two dense columns. I further extended the insight of column combining to support CNNs trained with power-of-two weights, where each weight is a single power-of-two term (e.g., 2^3) [25]. My recent work [37] has also applied this column combining methodology on both weight and input activations, and it demonstrates significantly hardware savings across a variety of DNN models and datasets while achieving the optimal accuracy performance.

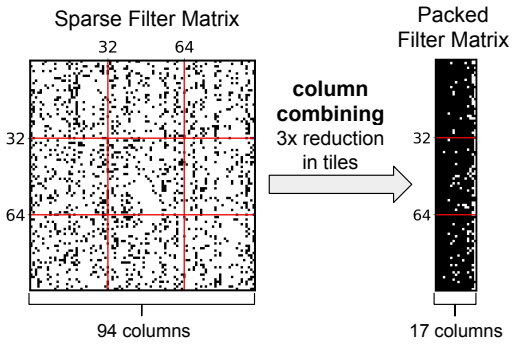

Figure 2: Column combining packs the 94 columns of a sparse matrix into 17 columns.

2.2 Group-based Quantization Scheme for Efficient DNN Operation [15, 36]

Term Quantization for Efficient DNN Inference [15]. The tolerance of deep learning models to small quantities of noise has led to a growing interest in designing various quantization approaches, which is proven to be effective in reducing the implementation cost of DNN Inference. Most of the previous works utilize the uniform quantization scheme for computing efficiency. However, uniform quantization can not precisely capture the bell-shaped distribution of weights and activations, further resulting in a larger quantization error and severe accuracy degeneration. To mitigate this issue, I propose a novel quantization technique called *term quantization* (TQ) [15]. Unlike conventional quantization that operates on individual values, TQ is a group-based method that keeps a fixed number of largest terms (nonzero bits in the binary representations) within a group of values (Figure 3). By exploiting normal-like weight and data distributions typically present in DNNs, TQ can achieve significantly smaller amount of quantization error over 8-bit uniform quantization while using only less than two terms (i.e., bits) to represent each weight and data value, which further allows for a great accuracy improvement over the conventional uniform quantization approach. With the real FPGA implementation, we demonstrated that term quantization can achieve $2-6\times$ better hardware efficiency over uniform quantization on multiple DNN models, and in the meanwhile achieving superior accuracy performance.

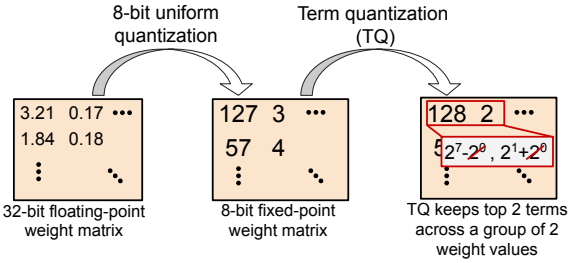

Figure 3: A 32-bit floating point weight matrix (left) is first quantized using uniform quantization with high precision (e.g., 8-bits). The resulting matrix (middle) are then further quantized via Term Quantization (right).

Multi-resolution DNN Inference with Term Quantization [36]. In general, there is an inherent trade-off between the classification accuracy and the precision of a DNN under a fixed quantization regime. Because of this trade-off, the DNN models are usually quantized under multiple resolutions (e.g., from 8-bit to 4-bit fixed-point bit-widths) to achieve varying degrees of performance/cost trade-off for practical hardware deployment. Given the superior performance of TQ described in [15], I proposed a novel multi-resolution term quantization scheme that efficiently supports quantization at multiple precisions at runtime [36]. Additionally, to support the term-based multi-resolution inference, I further developed a multi-resolution DNN training algorithm that jointly optimizes many sub-models across a wide range of resolutions. This multi-resolution DNN training algorithm will produce a multi-resolution model that can operate under different resolutions with an optimal accuracy performance. Based on the target hardware cost on executing DNN, the multi-resolution model can dynamically adjusting its resolution by simply changing the number of power-of-two terms on the weight and activation values, leading to an efficient performance/cost trade-off to suit the current hardware performance requirement. To implement the proposed term-quantized DNN, I further propose a novel multi-resolution Multiplier-Accumulator (mMAC) design that executes the value-level multiplication with term-based processing. Using real hardware implementations, I show that the mMAC design broadens the choices in trading off cost, efficiency, and latency across a wide range of computational budgets.

2.3 Efficient DNN Training Under Variable Precision Block Floating Point [35, 38]

While quantization has been shown to provide tremendous savings on hardware implementation costs for DNN inference, it can also be utilized to facilitate the DNN training process. Compared with the inference operations, DNN training requires extra computations to produce activation gradients and weight gradients, making the computing workload far exceed the inference. To mitigate the computing overhead of DNN training, I choose Block Floating-Point (BFP). Compared with floating-point and fix-point quantization schemes, BFP obtains a much wider dynamic range than fixed-point representation, while achieving a much lower implementation cost than floating-point representation. I further propose a Fast First, Accurate Second Training (FAST) system for DNNs, where the weights, activations, and gradients are all represented in BFP. FAST supports matrix multiplications with variable precision input operands, enabling incremental growth in DNN precision over the course of training. By using fast low-precision BFP in the early phase of the training, FAST can greatly shorten the training time while reducing the overall hardware resource usage. To implement FAST, I have designed a supporting hardware system with FAST multiplier-accumulator (fMAC) that operates on 3-bit chunks of mantissas across two groups of BFP numbers being multiplied. Additionally, in my recent work [38], I design a on-device training accelerator that uses eDRAM as the main on-chip memory and completely eliminates the need for off-chip communication. It has been proven that BFP quantization can also be applied to benefit the training implementation for the reversible DNN [5].

3 Multi-Agent Reinforcement Learning and Its Application

Many real-world applications (e.g. hardware chip placement, robotics control, autonomous driving) involve the participation of more than one single agent/player, and they can be modeled systematically as Multi-Agent Reinforcement Learning (MARL) problems. Specifically, cooperative MARL addresses the sequential decision-making problems of multiple autonomous agents that operate in a common environment, each of the agents aims to optimize a common goal by selecting the optimal action. To deploy MARL algorithm for practical usages, one of the major challenges is caused by the massive communication overhead among the MARL agents during the execution. To mitigate this problem, I have developed VBC [40] and TMC [41], which allow the MARL agents to operate under a minimum level of communication with a high level of robustness. Given the superior performance offered by VBC and TMC, I further adopt them to solve the practical problems including dynamic on-chip network reconfiguration [39] and client selection in Federated Learning [34].

3.1 Efficient Communication for Multi-agent Reinforcement Learning [40, 41]

In cooperative MARL, multiple agents interact with each other in a shared environment. Each agent only has access to the partial observations of the environment, and needs to make local decisions based on the partial observations to achieve the maximum team reward. Recent works [28, 4, 9] have demonstrated that introducing communications among the agents during execution can greatly improve the overall team performance. However, existing communication schemes often require agents to exchange an excessive number of messages, which results in a massive communication overhead and hinders its practicality in many real-world implementations. To mitigate this problem, I propose *Variance Based Control (VBC)* [40]. Under VBC, each agent first makes a preliminary decision based on its local information, and initiates the communication only when its confidence level on this preliminary decision is low. The confidence level is measured using the difference between the top two largest Q-values of the actions. Similarly, upon receiving the communication request, the agent replies to the request only when its message is informative. Our evaluation using a challenging set of StarCraft II benchmarks indicates that our method achieves $2-10\times$

lower in communication overhead than the state-of-the-art MARL algorithms, while allowing the agents to better collaborate by developing sophisticated strategies. Based on VBC, I further develop Temporal Message Control (TMC) [41], a simple yet effective approach for achieving succinct and robust communications in MARL. TMC allows the MARL agents to perform well under a highly lossy communication environment, while incurring an even lower communication overhead.

3.2 MARL applications [39, 34]

With the superior performances offered by VBC and TMC, I have applied them to facilitate the hardware system design. Specifically, my recent work has adopted VBC to dynamically reconfigure the on-chip network for the multi-core system, so that the resulting network topology can be customized for the applications that are currently being executed [39], further leading to a lower processing latency and power. I have also adopted VBC and TMC to perform run-time client selection for efficient Federated Learning (FL) implementations [34]. FL is a novel training paradigm that enables client devices to jointly learn a shared model by aggregating locally-computed models without exposing their raw data. We have shown that VBC can be utilized for client selection with optimal model accuracy, processing latency and communication efficiency.

4 Future Research

In their 2017 Turing Award lecture, Hennessy and Patterson dubbed this era as a "A New Golden Age for Computer Architecture" due in large part to the recent success of deep learning [8]. The high computational costs and noise tolerance of deep learning encourages fundamental changes to many aspects of the hardware architecture. On the other hand, the growing popularity of the DNN applications on resource-limited devices (e.g., wearable devices in AR/VR) will force the future DNN architectures to better suit the underlying hardware. Motivated by this perspective, I am excited to collaborate with researchers in the area of ML, architecture, VLSI, system, circuits, as well as engineers at technology companies, like Meta, Microsoft, and Intel to generate new ideas, tools, and methodologies for improving the energy efficiency and reliability of the next-generation DNN computing platform. Below I outline several directions that I am excited to pursue.

4.1 Hardware for Machine Learning

The tolerance of deep learning models to small quantities of noise has led to renewed interests in approximate computing paradigms. Generally, these approaches introduce some levels of noise in the computation for a significant increase in computational efficiency. I am interested in a full-stack approach that takes a unified view of approximate computing across all the algorithm level, compiler level and circuit level, as depicted in Figure 4. Specifically, I plan to explore efficient approximate computing schemes to facilitate the implementations of the emerging algorithms (e.g., Attention-based DNN model [23, 30], Dynamic DNN [29], etc) in various applications (e.g. deep learning inference, graph mining, etc). By leveraging their tolerance on the rounding noise, I will co-design the hardware platforms for the proposed algorithms by optimizing all the underlying arithmetic (e.g., term-based computation [15]), SoC architecture, and distributed system. I will also open my mind to explore the cross-disciplinary area between deep learning and other computing paradigms such as quantum computing, cryogenic computing and probabilistic computing. Finally, I am also interested in co-designing the compilation framework by improving the existing machine learning compiler (e.g., TVM [2]) to realize the maximum degree of efficiency, flexibility and generalizability across the full stack.

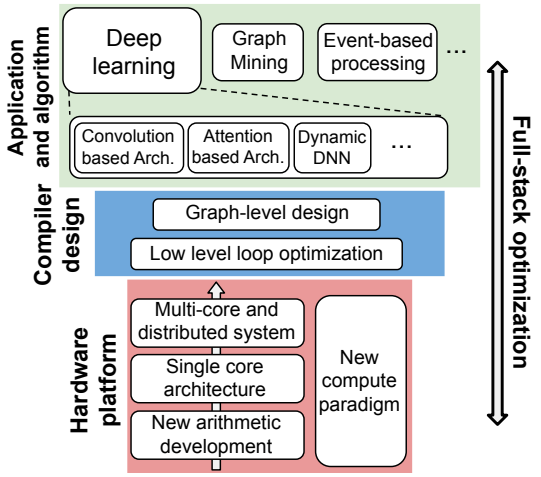

Figure 4: Overview of future work plan.

Recently, there is a growing demand on training DNN locally within the edge devices. For example, Federated Learning (FL) requires an on-device DNN training with local user data, which enables users to adapt the DNN model to their personal data and continuously improve their accuracy based on users' preference. Seeing this, I would like to explore the possibility of co-designing the DNN training algorithm, the training compiler, and the hardware architecture for efficient training processes. For example, applying term quantization to lower the implementation cost for training the attention-based models.

4.2 Machine Learning for Hardware

Multiple works have demonstrated the effectiveness of the AI algorithms on facilitating the hardware design. For example, a recent work by Google [26] posed chip floorplanning as a reinforcement learning problem, and relied on RL policy to optimize the quality of chip placements. They showed that the RL-based algorithm can

greatly outperform the current state-of-the-arts solution while obtaining significantly shorter design latency than the traditional EDA tools. The resulting floorplan solutions generated by the RL have been adopted in the product tape-out of a recent-generation Google tensor processing unit (TPU) accelerator. Other works have also demonstrated the effectiveness of RL on transistor sizing in VLSI [32], memory mapping of the data [13] and power management in multi-core system [11]. My computer architecture knowledge and my RL expertise make me well-suited to explore in this direction.

4.3 DNN and Hardware Co-Design for Event-based Vision

Event cameras are bio-inspired sensors that differ from a conventional frame cameras: Instead of capturing images at a fixed rate, they asynchronously measure per-pixel brightness changes, and output a stream of events that encode the time, location and sign of the brightness changes. Compared with the traditional cameras that capture the RGB images, event cameras offer higher temporal resolutions and dynamic ranges. However, the time-series nature of event-based camera inputs makes it difficult to be processed with the conventional DNN solutions of vision tasks (e.g., convolutional neural network, vision transformer). I will collaborate with the research teams from Meta Reality Lab to develop the novel DNN architectures as well as the supporting hardware platform for efficient event-based vision processing. The growing popularity on event-based vision will fundamentally shape the field of the DNN for vision tasks in the near future.

4.4 Practical Implementation of Multi-agent Reinforcement Learning

Although my previous research shows that it is possible for the MARL agents to achieve the same level of performance with much lower communication overhead, it has not yet delivered a complete solution to fully realize the MARL system in practice. Given the varying processing speeds and network conditions of the agents during the operation, one of the most important open problems in MARL is how to handle the straggler agents during the practical execution. The straggler MARL agents will severely degrade the overall system performance by keeping the fast agents waiting for the messages from them. To build an efficient MARL system, we will need to design an intelligent algorithm to detect and eliminate the messages from the straggler agents at runtime while maintaining a good team reward.

References

[1] Nvidia a100 tensor core gpu architecture. `https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf`.

[2] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, 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)*, pages 578–594, 2018.

[3] Xin Dong, Sai Qian Zhang, Ang Li, and HT Kung. Spherefed: Hyperspherical federated learning. In *European Conference on Computer Vision*, pages 165–184. Springer, 2022.

[4] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[5] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

[6] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] John L Hennessy and David A Patterson. A new golden age for computer architecture. *Communications of the ACM*, 62(2):48–60, 2019.

[9] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *arXiv preprint arXiv:1805.07733*, 2018.

[10] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark,

Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[11] Da-Cheng Juan and Diana Marculescu. Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 97–102, 2012.

[12] Sheng-Chun Kao, Geonhwa Jeong, and Tushar Krishna. Confuciux: Autonomous hardware resource assignment for dnn accelerators using reinforcement learning. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 622–636. IEEE, 2020.

[13] Shauharda Khadka, Estelle Aflalo, Mattias Marder, Avrech Ben-David, Santiago Miret, Shie Mannor, Tamir Hazan, Hanlin Tang, and Somdeb Majumdar. Optimizing memory placement using evolutionary graph reinforcement learning. *arXiv preprint arXiv:2007.07298*, 2020.

[14] H. T. Kung*, Bradley McDanel*, and Sai Qian Zhang*. Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization. *24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019 (equal contribution, names are ranked alphabetically).

[15] H. T. Kung*, Bradley McDanel*, and Sai Qian Zhang*. Term quantization: Furthering quantization at run time on quantized dnns. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020 (equal contribution, names are ranked alphabetically).

[16] Hsiang-Tsung Kung*, Bradley McDanel*, and Sai Qian Zhang*. Mapping systolic arrays onto 3d circuit structures: Accelerating convolutional neural network inference. In *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 330–336. IEEE, 2018 (equal contribution, names are ranked alphabetically).

[17] HT Kung, Bradley McDanel, and Sai Qian Zhang. Adaptive tiling: Applying fixed-size systolic arrays to sparse convolutional neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1006–1011. IEEE, 2018.

[18] HT Kung*, Bradley McDanel*, Sai Qian Zhang*, Xin Dong, and Chih Chiang Chen. Maestro: A memory-on-logic architecture for coordinated parallel use of many systolic arrays. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, volume 2160, pages 42–50. IEEE, 2019 (equal contribution, names are ranked alphabetically).

[19] HT Kung*, Bradley McDanel*, Sai Qian Zhang*, CT Wang, Jin Cai, CY Chen, Victor CY Chang, MF Chen, Jack YC Sun, and Douglas Yu. Systolic building block for logic-on-logic 3d-ic implementations of convolutional neural networks. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2019 (equal contribution, names are ranked alphabetically).

[20] Yuhang Li, Xin Dong, Sai Qian Zhang, Haoli Bai, Yuanpeng Chen, and Wei Wang. Rtn: Reparameterized ternary network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4780–4787, 2020.

[21] Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 62–68. IEEE, 2020.

[22] Ting-Ru Lin, Drew Penney, Massoud Pedram, and Lizhong Chen. A deep reinforcement learning framework for architectural exploration: A routerless noc case study. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 99–110. IEEE, 2020.

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[24] Bradley McDanel, HT Kung, and Sai Qian Zhang. Saturation rram leveraging bit-level sparsity resulting from term quantization. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021.

[25] Bradley McDanel*, Sai Qian Zhang*, H. T. Kung, and Xin Dong. Full-stack optimization for accelerating cnns using powers-of-two weights with fpga validation. *International Conference on Supercomputing*, 2019 (equal contribution, names are ranked alphabetically).

[26] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[28] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[29] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[30] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[32] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[33] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. Adaptive distributed convolutional neural network inference at the network edge with adcnn. In *49th International Conference on Parallel Processing-ICPP*, pages 1–11, 2020.

[34] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9091–9099, 2022.

[35] Sai Qian Zhang, Bradley McDanel, and HT Kung. Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 846–860. IEEE, 2022.

[36] Sai Qian Zhang, Bradley McDanel, HT Kung, and Xin Dong. Training for multi-resolution inference using reusable quantization terms. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 845–860, 2021.

[37] Sai Qian Zhang, Tambe Thierry, McDanel Bradley, Dong Xin, Jeff Jun Zhang, Gu-Yeon Wei, and David Brooks. Accelerating dnn execution with adaptive n:m pruning on both weight and data. *In submission*, 2023.

[38] Sai Qian Zhang, Tambe Thierry, Nestor Cuevas, Jeff Jun Zhang, Gu-Yeon Wei, and David Brooks. Camel: Co-designing ai models and embedded drams for efficient on-device learning. *In submission*, 2023.

[39] Sai Qian Zhang, Tambe Thierry, Gu-Yeon Wei, and David Brooks. A multi-agent reinforcement learning approach for reconfigurable network-on-chip routing. *In submission*, 2023.

[40] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. *Advances in Neural Information Processing Systems*, 32:3235–3244, 2019.

[41] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and robust multi-agent communication with temporal message control. *Advances in Neural Information Processing Systems*, 33, 2020.