Research Statement

Jeff (Jun) Zhang (jeffzhang@seas.harvard.edu)

For decades, the semiconductor industry has benefited from the technology scaling (i.e., Moore's law and Dennard scaling) to make computer chips smaller, faster, and more energy efficient, allowing general-purpose central processing units (i.e., CPUs) to handle most of our computing needs. However, higher cost (e.g., yield, reliability and process variation issues), increasing power density, and fundamental physical constraints have become showstoppers for continued scaling; at the same time, emerging applications like deep learning, AR/VR, and autonomous driving are increasingly compute-intensive and power-hungry. This has resulted in the adoption of application-specific computing, in turn creating exciting new research challenges for the hardware industry.

My primary research interests are at the intersection of *machine learning (e.g., deep learning)*, *computer architecture*, *electronic design automation (EDA)*, and *embedded systems*. **The overarching goal of my research is to achieve energy-efficient and robust domain-specific computing: from circuits to architectures and systems.** I believe in rigorous evaluations via extensive hardware prototypes (including recent chip tape-outs [1]) and in extensive government/industry collaborations; during my Ph.D. I interned with *Samsung Semiconductor* and *Microsoft Research*. At Harvard, I lead our ongoing collaborations with *Pacific Northwest National Laboratory*, *Intel Labs*, and *IBM Research*. The outcomes of my research have been published at top-tier architecture, EDA, testing, and embedded systems venues (MICRO, DAC, DATE, ICCAD, ESWeek, VTS, etc.), and my research internships have resulted in two U.S. patents. My thesis was short-listed for *best presentation award* at DATE Ph.D. Forum and was the semifinalist in the TTTC's E.J. McCluskey Doctoral Thesis Competition.

Energy-Efficient and Reliable Deep Learning Acceleration

Modern System-on-Chip (SoC) designers have embraced custom hardware acceleration to improve energy efficiency by tailoring the architecture to the characteristics of a particular application. Custom accelerators eliminate unnecessary operations needed for general-purpose processors, incorporate more effective use of the memory hierarchy, and exploit much finer-grained parallelism. Recent examples include Google's Tensor Processing Unit (TPU), which utilizes a sizeable **systolic array (SA)**, a tightly-coupled grid of multiply-and-accumulate (MAC) units at its core to speed up matrix multiplication (or convolutions) for deep neural network (DNN) workloads, and achieves a 30x - 80x higher performance/Watt than CPU or GPU based solutions; the efficiency comes from greater amortization of memory accesses and the obviation of complex routing between MAC units in the SA.

Enabling Energy-efficient and Timing Error Resilient Deep Learning at Scale

The Google TPU was released as I was beginning my Ph.D. research. The first question I sought to ask was: *can we make the TPU more energy-efficient?* To this end, I proposed the **ThunderVolt** architecture, a SA design with architectural support for voltage underscaling based timing speculation that reduces TPU-like deep learning accelerators' energy consumption by 2x without hurting *throughput* and *classification accuracy.* Timing speculation (TS) [2] is a powerful technique, first studied by Ernst et al. in 2003 (and a 2021 MICRO Test-of-Time awardee) that proposes to under-scale a chip's supply voltage to reduce power consumption. At the expense of occasional timing errors, which are caught by a specially designed "Razor" flip-flop (FF), TS recovers the correct computation from Razor FFs by stalling the processor's pipeline stages. We refer to this as *timing error detection and recovery (TED).* The premise of TED is that as long as the timing error rate remains low, the overall energy efficiency improves.

Given the past success of TED schemes, an obvious question is whether it can be directly applied to increase the energy efficiency of a TPU? However, I noted that, although the simplicity of SA's micro-architecture makes it easy to scale, it requires a *strict* schedule for all its MAC units to work correctly. That is, any MAC operation (e.g., TPUv1 has 65,536 MAC units) that goes out-of-sync will nullify the whole array's computation. Under

[1]Our DARPA-funded Domain-Specific System on Chip (DSSoC) project *EPOCHS* [1] is IBM-led, multi-university (Harvard/Columbia/UIUC) effort on the future autonomous driving system. We successfully taped out two 12 *nm* DSSoC chips in October 2020 and October 2021, respectively. My contributions include accelerator system-level integration, SoC test driver design, VLSI design flow, gate-level testing and verification, and chip bring-up.

this constraint, a conventional timing speculation scheme such as TED is not a viable option anymore — TED would have to stall the whole array to re-execute/recover any single MAC operation with timing errors. Our detailed gate-level simulation on a 256x256 SA empirically verifies this: timing error rates for large SA under the TED scheme are very high, so the benefits from TS are not significant. My second idea is based on the conventional wisdom which suggests that DNNs are inherently error resilient to small bit-flips. If TED incurs too much re-execute overheads, *can we simply let the timing error propagate (TEP)*? Unfortunately, I empirically showed that, **for the first time**, few timing errors are bad enough to crash the DNN's classification accuracy. In fact, timing errors for arithmetic operations typically occur in higher order bits, resulting in potentially large changes in the value of the partial sum computed by the MAC unit. Instead, these two attempts motivate a new timing error recovery technique, **ThunderVolt** [4, 5]. ThunderVolt is an innovative circuit and micro-architectural solution to drop (or zero out) faulty MAC operations, enabling voltage underscaling TS without re-execution. Empirically, we observe that ThunderVolt allows to aggressively under-scale the supply voltage of DNN accelerators (with up to 10% timing errors) without hurting accuracy.

A major roadblock in the ThunderVolt study is that running detailed gate-level timing simulations (GLS) to predict the timing error behavior of digital logic is prohibitively time-consuming. Since they preserve the view that most closest to real silicon, GLSs are orders of magnitude slower than functional simulations. For large DNN accelerators and fast-growing deep learning models, full GLS will quickly be infeasible. In follow-up work, we mitigate this challenge with **FATE** [6], a new EDA tool that facilitates timing error simulation, enabling quick yet accurate exploration of different micro-architecture design choices for timing speculative DNN acceleration. With abundant ground-truth data obtained in ThunderVolt, I leveraged the powerful deep learning techniques to estimate the delay of any MAC computation accurately. Given the activation inputs are shared among all columns of the MAC array, I also proposed to statistical sample only a subset of MAC units for timing simulations and probabilistically inject errors to the remaining MACs. Our evaluation show that FATE provides more than two orders of magnitude speed-up against full detailed GLS, while introducing less than 6.2% average prediction error on the timing error behavior.

ThunderVolt and FATE have been cited across top-tier venues in both the circuits and architecture communities, and covered in the popular press.

Analyzing and Mitigating the Impact of Permanent Faults on DNN Accelerators

Another important challenge with advanced nano-meter CMOS technology scaling is increased fault rates, including both permanent (hard errors) and temporary faults (soft errors). (As a side note, we noticed several such faults in the recent 12 *nm* DSSoC tape-out.) Temporary faults might occasionally impact the DNN's classifications, but their overall impact on classification accuracy is restricted, even at high soft error rates. However, permanent faults can affect the computation of *every* DNN execution and significantly reduce the classification accuracy, as we show **for the first time**, on a large TPU-like SA [7]. These issues will be further exacerbated for emerging wafer-scale computing platforms like Cerebras. Unfortunately, redundancy based fault-tolerance solutions [8, 9] used in general-purpose computing incur high resource and energy overheads for accelerator-rich SoCs.

Domain-specific computing opens up new fault-tolerance opportunities that are *lightweight, customized* to the application and the underlying hardware. For example, systolic array based DNN accelerators often use *weight-stationary* dataflow; and permanent faults, at least those that are related to manufacturing defects, can be identified and located during post-fabrication testing. This observation motivates the design of **fault-aware pruning (FAP)** for systolic arrays with manufacturing defects [7, 10]. FAP leverages the *static* one-to-one mapping between the DNN *weights* and MAC units in a *weight-stationary* SA. With a fault map that tells where the MACs with defects are, DNN *weights* that will be mapped to those MACs during execution are also known. At run-time, FAP implements a simple, low-area overhead circuitry to bypass the faulty MACs' computation, mitigating their impact on the final partial sum of the SA. Further, the identified weights are *pruned* (set to zero) away, and the remaining weights of the DNN are fine-tuned by incremental re-training to recover classification accuracy, which we refer to as **FAP+T**. FAP+T incurs extra training cost for each chip that may have different defects, but the one-time cost is only in *test time*, and can be amortized by over the chip's lifetime. Empirically, I found that FAP and FAP+T enable DNN accelerators to be used even at relatively high permanent fault rates with only marginal accuracy loss. In the follow-up work [10], I investigated how different weights-to-SA mapping strategies can help improve fault tolerance. I also showed that FAP and FAP+T could be adapted to repair process variation-induced permanent faults.

FAP received the **best paper award nomination** at *IEEE VLSI Test Symposium (VTS)*. It is also listed as the most frequently accessed documents as recorded by *IEEE Design & Test*. FAP introduced reliability as a serious concern for DNN acceleration and opened up low-cost fault tolerance opportunities for domain-specific applications. It motivated the work of several research groups on advanced fault mitigation strategies for deep learning hardware.

Reducing On-chip Memory Footprint for Efficient Low-power Embedded DNN Acceleration

During my research internship at the *Microsoft HoloLens team*, I observed that for embedded systolic accelerators running computer vision models, on-chip memory (SRAM) used to buffer DNN's activation inputs (or feature maps) is a significant contributor to total on-chip energy consumption. There are three main reasons: (1) computer vision models generate a large size of activation data; (2) strict power budget on embedded devices could only afford small MAC array (e.g., 64x64 SA), thus the activation memory access increases (due to fewer data reuse); (3) larger capacity is provisioned for on-chip buffers as a design philosophy to amortize the more expensive off-chip data communication.

To address this, I designed **CompAct** [11], **the first** SA-based architecture that enables on-chip compression of DNN activations. An immediate challenge for CompAct is the choice of the compression scheme. I found that although sophisticated entropy-based encoding might provide a better compression ratio, the decoding process of variable-length codewords is hard to meet SA's data throughput requirement. While a fixed-length encoding scheme such as run-length-coding (RLC) is fast to decode, it requires a regular data access pattern to ease the encoding. In CompAct, I discovered a particular DNN activation schedule that makes RLC work on SA, without performance degradation. I also proposed a lossy RLC scheme with principled approximation on the activation data to improve the compression ratio. The decoding logic of CompAct only fetches the on-chip buffer once for each compressed data word, so during the idle cycles, the leakage power of activation buffer can be further optimized. Building with these ideas, CompAct is able to achieve more than 2x energy reduction on the activation buffer of the SA-based accelerator. I also showed that CompAct could work synergetically with DNN pruning, a widely used algorithmic technique that reduces the DNN's memory footprint. The design of CompAct has generated two U.S. patents [12, 13] and shed light on Microsoft's next generation of DNN hardware.

Designing Higher-Quality Machine-Learning-as-a-Service (MLaaS)

At the system level, machine learning (ML) based prediction models, and especially DNNs, are being increasingly deployed as cloud services to provide inference serving (or predictions) for a range of applications, such as computer vision, natural language processing, and recommendation. Like other cloud services, MLaaS has the quality of service (QoS) requirements specified as service level agreements (SLAs) between the users and the cloud provider. Typical SLAs define the expected response time (i.e., tail latency), throughput and availability, etc. For ML, in particular, the **prediction quality** of the service should also be a critical metric but has not been encapsulated in SLAs for traditional applications.

In a recent collaboration with *Microsoft Research*, I characterized a state-of-the-art MLaaS framework (Berkeley's **Clipper**) on Azure Virtual Machines and found several issues in the presence of a highly dynamic, fluctuating load. Specifically, existing MLaaS systems either drop deadline-missed requests or significantly expand hardware resources in response to load spikes. In our study [14], we showed that *for DNNs, a diverse model architecture can be pre-trained for the same application; when a specific model is being deployed, computing resource allocation on data-level vs. model-level parallelism also plays a role in the tail latency.* Model diversity and different resource allocation strategies carve out a widespread accuracy vs. latency Pareto frontier, opening new opportunities to deal with load fluctuation. Specifically, I proposed *effective accuracy*, a new QoS metric that quantifies users' expectation on *correctly predicted requests within the deadline.* Guided by this new metric, an online **Model-Switching** engine is designed and implemented within Clipper. During real-time inferencing, Model-Switching is able to switch to the best model quickly in response to load spikes, providing high-quality service without the need for over-provisioning the hardware capacity.

During my post-doctoral research [15], we (with *Facebook AI Research*) aim to optimize deep learning based **recommendation serving**, a specific MLaaS that constitutes an overwhelming fraction of machine learning cycles in production data centers (e.g., Facebook, Google). Facebook's data show that production-scale recommendation model sizes grow rapidly (> 10x) in just three years, resulting in a substantial increase in infrastructure demands. *For recommendation service, the end-to-end prediction quality depends not only on the accuracy of the model but also on the number of ranked items.* Ranking all candidate items with the most accurate and complex models is superfluous because eventually, only a small portion of the items will be recommended to individual users. Based on this key insight, we proposed **RecPipe**, a multi-stage recommendation engine that has two components: (1) a light-weight, less accurate frontend model that coarsely filters a large set of candidate items; (2) a heavier, more accurate back-end model refines ranking only on top candidates. We showed that at iso-quality, RecPipe's multi-stage ranking can greatly reduce overall compute demand and embedding access compared to the single-stage recommendation. Furthermore, we found that running multi-stage ranking on exiting heterogeneous computing platforms (e.g., CPU-GPU) and the state-of-the-art data center accelerators (e.g., TPU) results in costly inter-stage data transfer and low resource utilization across stages. For these reasons, I designed and implemented an on-chip filtering unit that eliminates host-accelerator communication between recommendation stages; and a run-time reconfigurable systolic array that simultaneously processes multi-stage queries and exploits data- and model-parallelism at each stage respectively. Overall, RecPipe improves tail-latency by up to 5x and throughput by up to 10x at iso-quality.

RecPipe participated in the first MICRO 2021 Artifact Evaluation and received all three badges (Artifact Available, Functional, Reproduced). We also open-sourced Model-Switching and RecPipe with *Intel Labs*, and are working together to design a better multi-stage model-switching recommendation service system.

Future Research Agenda

In future research, I am excited to collaborate with researchers in the area of ML, system, architecture, VLSI circuits, as well as engineers at tech companies, like *Facebook*, *Microsoft*, *IBM*, and *Intel* to generate new ideas, tools, and methodologies for improving the energy efficiency and reliability of the next-generation computing platform. Below I outline several directions that I am excited to pursue:

Efficient and Resilient Hardware/Software Co-design for Learning based Autonomous System. Real-world autonomous systems are composed of a complex computational pipeline that holistically accomplishes the mission of the domain. For example, an autonomous machine navigates in an environment may need perception/sensing (with a camera, GPS, depth sensor, etc.), localization and mapping, communication, and planning and control. Recent advances in deep learning have empowered many of these sub-tasks, opening up new challenges and co-design opportunities for hardware. In ongoing work [16], we have recently investigated AI Habitat, Facebook's physically realistic simulation platform for Embodied AI research. Specifically, my preliminary profiling shows that each sub-module of the robot has its unique compute and memory demands for navigation tasks, suggesting a careful architectural design under the power, performance, and area constraints. One interesting question could be: *should we design many fixed-function accelerators that map to each sub-module of the complex system, or how beneficial is programmability in domain-specific architectures?* To improve the utilization of the on-chip resources, an efficient scheduler and run-time resource management are also worthy of further research. In addition, a learning-based system can offer greater efficiency by various co-design techniques across the computing stack, such as DNN pruning and compression, reduced precision and quantization, and approximate computing. I plan to evaluate this further by discovering the trade-off between every single module's accuracy and the overall system's efficiency.

The reliability of autonomous vehicles is another critical topic. Unlike single ML-based applications that use simple prediction accuracy as the metric, it is crucial to analyze the impact of faults vertically (across the stack) and horizontally (end-to-end) for the complex system. I believe identifying the correct metric to evaluate the system's fault tolerance is the way to design efficient application-specific resilient techniques. At Harvard, I made key contributions to the DARPA-funded DSSoC project for the future autonomous driving system. Our chip tape-outs set up an invaluable stage to answer the above research questions and motivate a new set of practical innovations.

Agile Hardware Design and Integration, Robust Design-time and Run-time Optimization. The complexity and heterogeneity in large SoCs stress the traditional "waterfall" hardware development process.

DSSoC deserves a better design automation tool to reduce the design time and cost. For this reason, I'm currently working with researchers at *Pacific Northwest National Laboratory* to design an open-source domain-specific application-to-hardware synthesizer based on Google's Multi-Level Intermediate Representation (MLIR) infrastructure [17, 18]. I'm also an external contributor to Columbia's open-source heterogeneous SoC platform (ESP) [2], which our own DSSoC tape-outs relied on. In the future, I plan to continue building more comprehensive EDA tools to support emerging technologies and applications. For example, advanced 2.5D packaging technologies enable multi-chip-module (MCM) integration, where smaller chiplets can be connected together post-fabrication via low-latency and high-bandwidth interconnects. However, MCM integration also comes with an enormous design space challenge: *How to optimally select which chiplets to build up for a given application?* Other emerging technologies include: embedded nonvolatile memory (eNVM) with multi-level cell (MLC) is a promising alternative storage solution for large deep learning models; Processing-in-Memory (PIM) is another attractive solution for deep learning since it reduces the need for data movement and can perform several fundamental deep learning operations such as dot products and non-linear activations at a low cost. However, both MLCs and PIM (e.g., ReRAM devices) are inherently noisy and error-prone, requiring a principled tool to investigate. Furthermore, various run-time techniques can enable *better-than-worst-case* design opportunities that jointly optimize accuracy, performance, energy-efficiency, and reliability for ML-based applications, such as DNNs [19] and hyperdimensional computing [20]. Together, I foresee an *expended* co-design space at both design-time and run-time to explore where traditional heuristics may be sub-optimal. My prior experience applying mathematical optimizations (e.g., linear programming, dynamic programming) [21, 22] and ML methods (e.g., reinforcement learning) [23, 24] to design and optimize multiple computing systems will help tackle some of these challenges more efficiently.

MLaaS vs. On-device Personalized AI. With the increasing research and development efforts on domain-specific hardware, data centers nowadays have a heterogeneous fleet of backend computing resources, e.g., CPUs, GPUs, and accelerators. On the other hand, MLaaS at the data center adopts different types of ML models for various application tasks. These ML workloads show distinct memory-bandwidth or compute-bound characteristics across their DNN layers and have different SLA targets (e.g., tens to hundreds of milliseconds latency requirement). MLaaS also needs to serve billions of users with fluctuating capacity demand over time. Therefore, the availability and flexibility of computing resources are important. Serving a mixed ML workload with multi-tenancy can improve resource utilization but is also challenging in practice. To this end, I plan to design a better system infrastructure for MLaaS — that is self-aware and can adapt its workload scheduling/mapping and resource allocation strategies dynamically in response to the environment change without violating the SLAs.

To improve the service quality, data center often scales up the ML model size and performs frequent incremental training on newly generated data samples daily or weekly. However, training large ML models generates an enormous carbon footprint which is not sustainable. Moreover, a centralized model trained on the aggregated user data suffers from the long tail accuracy issue — i.e., long-tailed training samples from minorities induce the model bias. I envision over billions of ubiquitous mobile devices around the world unlock new opportunities for *personalized* AI. Running ML on the device can improve user experience with reduced service response time, make the service less dependent on network connectivity, and, more importantly, enable access to abundant private, contextual features only available locally. In the long term, I plan to look into *the system support for device-cloud collaborative learning*, where the cloud maintains a general centralized model, and at the same time, end-users have their customized models trained cooperatively with the cloud for better generalization. Each end user's model architecture can be tailored to the unique capabilities of their own mobile devices, and be trained over the local data, achieving deep personalization. I will devise system-level solutions that account for the performance variability of the device, network bandwidth and connectivity, and improve the overall energy efficiency of the personalized collaborative learning.

[2]`https://github.com/sld-columbia/esp/blob/master/CREDITS.md`

References

[1] Pradip Bose, Augusto Vega, Sarita Adve, Vikram Adve, Sasa Misailovic, Luca Carloni, Ken Shepard, David Brooks, Vijay Janapa Reddi, and Gu-Yeon Wei, "Secure and resilient SoCs for autonomous vehicles", *International Workshop on Domain Specific System Architecture (DOSSA), in conjunction with IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021. (Project performers are listed in the acknowledgement section.)

[2] Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler, David Blaauw, Todd Austin, Krisztian Flautner, and Trevor Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation", *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2003.

[3] Atif Yasin, Jeff Zhang, Hu Chen, Siddharth Garg, Sanghamitra Roy, and Koushik Chakraborty, "Synergistic timing speculation for multi-threaded programs", *IEEE/ACM Annual Design Automation Conference (DAC)*, 2016.

[4] Jeff Zhang, Kartheek Rangineni, Zahra Ghodsi, and Siddharth Garg, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators", *IEEE/ACM Annual Design Automation Conference (DAC)*, 2018.

[5] Jeff Zhang, Zahra Ghodsi, Kartheek Rangineni, and Siddharth Garg, "Enabling timing error resilience for low-power systolic-array based deep learning accelerators", *IEEE Design & Test*, 2019.

[6] Jeff Zhang, Siddharth Garg, "FATE: fast and accurate timing error prediction framework for low power DNN accelerator design", *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018.

[7] Jeff Zhang, Tianyu Gu, Kanad Basu, and Siddharth Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator", *IEEE 36th VLSI Test Symposium (VTS)*, 2018.

[8] Jeff Zhang, Edwin HM Sha, Qingfeng Zhuge, Juan Yi, and Kaijie Wu, "Efficient fault-tolerant scheduling on multiprocessor systems via replication and deallocation", *International Journal of Embedded Systems*, 2014.

[9] Xiaotong Cui, Jeff Zhang, Kaijie Wu, and Edwin HM Sha, "Efficient feasibility analysis of DAG scheduling with real-time constraints in the presence of faults", *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[10] Jeff Zhang, Kanad Basu, and Siddharth Garg, "Fault-tolerant systolic array based accelerators for deep neural network execution", *IEEE Design & Test*, 2019.

[11] Jeff Zhang, Parul Raj, Shuayb Zarar, Amol Ambardekar, and Siddharth Garg, "CompAct: on-chip compression of activations for low power systolic array based CNN acceleration", *ACM Transactions on Embedded Computing Systems (TECS)*, 2019.

[12] Amol Ambardekar, Shuayb Zarar, Jeff Zhang, "Selectively controlling memory power for schedule computations", *US Patent.*

[13] Shuayb Zarar, Amol Ambardekar, Jeff Zhang, "Compression-encoding scheduled inputs for matrix computations", *US Patent.*

[14] Jeff Zhang, Sameh Elnikety, Shuayb Zarar, Atul Gupta, and Siddharth Garg, "Model-switching: dealing with fluctuating workloads in MLaaS systems", *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2020.

[15] Udit Gupta, Samuel Hsia, Jeff Zhang, Mark Wilkening, Javin Pombra, Hsien-Hsin S Lee, Gu-Yeon Wei, Carole-Jean Wu, and David Brooks, "RecPipe: co-designing models and hardware to jointly optimize recommendation quality and performance", *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2021.

[16] Mingsheng Yin, Akshaj Veldanda, Amee Trivedi, Jeff Zhang, Kai Pfeiffer, Yaqi Hu, Siddharth Garg, Elza Erkip, Ludovic Righetti, and Sundeep Rangan, "Millimeter wave wireless-assisted robotic navigation with link state classification", *arXiv preprint, Under Review*, 2021.

[17] Jeff Zhang, Nicolas Bohm Agostini, Shihao Song, Cheng Tan, Ankur Limaye, Vinay Amatya, Joseph Manzano, Marco Minutoli, Vito Giovanni Castellana, Antonino Tumeo, Gu-Yeon Wei, and David Brooks, "Towards automatic and agile AI/ML accelerator design with end-to-end synthesis", *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2021.

[18] Cheng Tan, Nicolas Bohm Agostini, Jeff Zhang, Marco Minutoli, Vito Giovanni Castellana, Chenhao Xie, Tong Geng, Ang Li, Kevin Barker, and Antonino Tumeo, "OpenCGRA: democratizing coarse-grained reconfigurable arrays", *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2021.

[19] Jeff Zhang, Kang Liu, Faiq Khalid, Muhammad Abdullah Hanif, Semeen Rehman, Theocharis Theocharides, Alessandro Artussi, Muhammad Shafique, and Siddharth Garg, "Building robust machine learning systems: Current progress, research challenges, and opportunities", *IEEE/ACM Annual Design Automation Conference (DAC)*, 2016.

[20] Sizhe Zhang, Ruixuan Wang, Dongning Ma, Jeff Zhang, Xunzhao Yin and Xun Jiao, "Energy-efficient brain-inspired hyperdimensional computing using voltage scaling", *ACM/IEEE Design, Automation and Test in Europe (DATE)*, 2022.

[21] Jeff Zhang, Tan Deng, Qiuyan Gao, Qingfeng Zhuge, and Edwin HM Sha, "Optimizing data allocation for loops on embedded systems with scratch-pad memory", *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2012.

[22] Yan Wang, Kenli Li, Jeff Zhang, and Keqin Li, "Energy optimization for data allocation with hybrid SRAM+ NVM SPM", *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2017.

[23] Jeff Zhang, Siddharth Garg, "BandiTS: dynamic timing speculation using multi-armed bandit based optimization", *ACM/IEEE Design, Automation and Test in Europe (DATE)*, 2017.

[24] Wen Zhang, Jeff Zhang , Mimi Xie, Tao Liu, Wenlu Wang and Chen Pan, "M2M-Routing: environmental adaptive multi-agent reinforcement learning based multi-hop routing policy for self-powered IoT systems", *ACM/IEEE Design, Automation and Test in Europe (DATE)*, 2022.