Heterogeneous Computing in Machine Learning

Edge AI, heterogeneous processors and homogenous APIs

COMP6281 Literature Survey

Riley Ballachay

Introduction

Practically every sector of industry has felt the impact of artificial intelligence (AI) over the past decade. AI is being used to optimize processes, enhance customer experience and develop new products[1]. The hardware supporting these algorithms has had to make significant improvements in latency and throughput to keep pace. One domain of hardware which has rapidly changed in recent years to accommodate AI is the edge; a paradigm in which computations are performed at the network's edge, typically on-device, as opposed to a centralized data center. Edge AI offers an attractive solution to security, high latency and connectivity issues that can plague dependency on centralized communication for processing. The result is a faster, more reliable and more secure processor for AI deployment. The global edge AI market is expected to expand at a rate of 21% over the next 7 years, a near quadrupling of the market in under ten years [2]. This growth can be attributed to the expansion of digitalization in real-time applications including energy, healthcare and manufacturing. To accommodate the unique constraints of edge environments, similarly unique processors have been designed. While GPUs, the most-used processor for machine learning (ML), existed prior to the emergence of deep learning, others were created specifically for deep learning. Figure 1 shows a brief comparison of processors and their unique features.

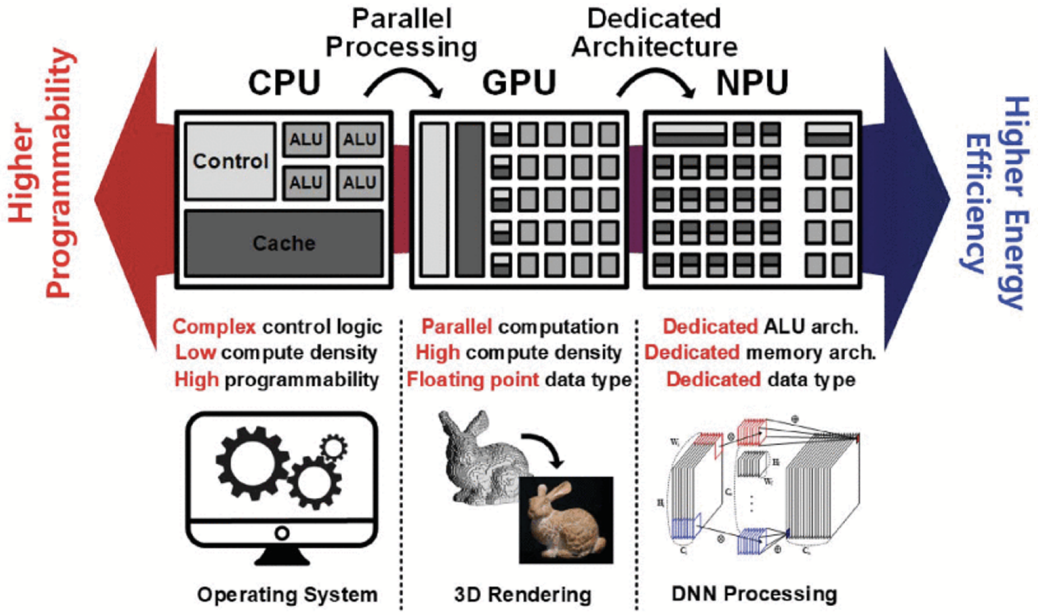

Figure 1: Comparison of processors: CPU, GPU and DPU. Adapted from Yoo, 2020 [3].

Machine learning, specifically deep learning, requires millions of individual computations that can be exponentially accelerated by parallelization. Accelerators like GPUs are tailored to these applications - a gap of up to seven-fold is theoretically possible between CPU and GP. In addition to the hardware, the creation of standardized tools and languages has permitted for a wide adoption of GPU programming including CUDA and OpenCL [4]. All major GPU programming API's treat the CPU as host and GPU as a target device, awaiting execution of

GPU instructions while the CPU remains idle. This is largely due to the separation of processors over the slow PCIe bus. To address these limitations, there has been a rapid increase in the development of heterogeneous computing systems (HCS), featuring unified memory spaces. Such systems are abundant in edge devices, including Apple Silicon and NVIDIA Jeston [7]. Programmers are transitioning from the CPU vs GPU debate toward a solution of combining the two to achieve even greater performance. The technology in general is still in its infancy, however, and is in need of good abstractions and unified programming interfaces [9]. This opens up great opportunities for engineers and enterprises to propose innovative solutions.

Edge AI

With recent innovations in artificial intelligence efficiency and the proliferation of internet of things and mobile devices, we have seen a rapid transition of AI computation from centralized locations to the edge - a technology paradigm where computation is performed at the network ‘edge’ as opposed to centralized data centers. Examples of edge devices include mobile phones, smart sensors and routers. Edge solutions are cost effective, more responsive and reliable and more secure than cloud equivalents. Artificial intelligence (AI) stands to benefit greatly from these features of edge technology. Nearly every industry will be impacted by artificial intelligence in the coming years, and edge AI seems well poised to lead that charge. The international data corporation predicts that by 2025, there will be 150 billion intelligent edge devices in use; the industry as a whole is expected to balloon from $3.5 billion to $43.4 billion by 2027 [1]. Deep learning, the subset of machine learning models which require high amounts of computation, is expected to be an integral part of edge devices moving forward. It is thus expected that edge AI will be a rapidly growing field, and understanding how models work and optimizing their performance will be an important priority for engineers.

Deep Learning Processors

Traditional hardware solutions used in data centers have been demonstrated unsatisfactory for edge AI applications. Their low energy efficiency and large size do not satisfy the unique requirements of edge devices. To meet these requirements, new high-performance silicon processors and integrated systems have been developed. The transformation of Graphical Processing Units (GPUs) into general-purpose parallel processors has been one significant part of meeting these demands. GPUs are particularly popular for rapid training and inference of deep learning models. Mathematical operations like matrix multiplications that underpin deep learning are highly parallelizable operations, meaning that operations can be distributed over the many cores of a GPU. The quantity of independent cores in GPUs is important, as a CPU with eight cores will be an order of magnitude slower in parallelizable operations than a GPU with thousands. Figure 2 shows the speed of deep learning inference of varying batch size on CPU versus GPU architecture. As the batch size increases, the GPU becomes markedly faster, as the parallelization of process increases [10]. Specialized deep learning accelerators including

Neural Processing Units (NPUs) and Deep Learning Processing Units (DPUs), have also been created, which feature all the qualities of GPUs with particular traits tailored to ML. It is common for embedded devices to come equipped with CPU, GPU and NPU.

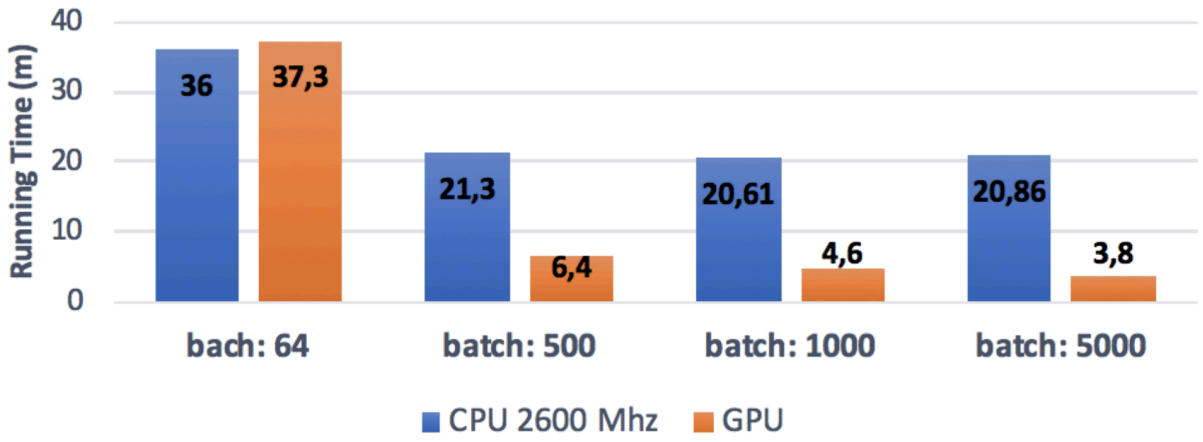

Figure 2: Comparison of inference time versus batch size for GPU vs CPU. Adapted from Buber, E. & Diri, B. (2018) [10]

Deep Learning Frameworks

The two most popular frameworks for deep learning are Tensorflow and PyTorch [4]. These frameworks have their respective subprojects Tensorflow Life and PyTorch Mobile for edge applications. These projects are highly optimized for running on GPUs or other specialized deep learning hardware. TensorRT, a similar SDK for high-performance inference from NVIDIA, is built upon the GPU programming language CUDA is of particular prominence in the industry. It enables the user to speed up inference using a litany of optimization techniques. However, like many other SDKs, TensorRT assumes inference is undertaken on a single processor - either the CPU, GPU or NPU, never both [11]. To understand why, it is important to understand the programming languages used for GPU inference. CUDA, OpenCL and DirectCompute are the most common languages for GPU programming. They are general-purpose languages that allow for abstraction and easy development on GPU devices. Execution of GPU programs written using these languages typically runs as follows: (1) Compile the GPU kernel (GPU code), (2) Read input, (3) Copy data to GPU, (4) Run kernel on GPU, (5) Copy data back to CPU and (6) Process returned data using the CPU [12]. Because of high latency in transfer between CPU and GPU, programs typically transfer data once, and as a result, the CPU ends up being idle while GPU kernels are running. This is because CUDA (or OpenCL etc) programs are written explicitly for the GPU, not for the system. This is a trend observed throughout the programming world: frameworks are written for one of the two processors - MPI, POSIX Threads and OpenMP for CPU and OpenCL and CUDA for GPU.

The Growth in Heterogeneous Computing Systems

Trends in high performance computing have shown a preference for a new type of system in previous years. Heterogeneous computer systems, instead of two discrete CPU and GPU connected by a bus or board, utilize shared memory in order to entirely eliminate data transfer times. Removing such data transfer bottlenecks has shown to improve performance by up to 30 times in data centers [7]. Heterogeneous computing has also become very popular on the edge, particularly in deep learning-applications. NVIDIA has a series of edge devices, Jetson, which typically contain unified shared memory between a CPU and GPU and multiple additional deep learning accelerators. Another prominent example is Apple's line of System-on-a-Chip (SoC) processors, which use unified CPU/GPU memory and an additional Apple Neural Engine (ANE). Apple Silicon has the additional benefit of the Core ML framework, a heterogeneous computing SDK for deep learning that automatically alternates between CPU and GPU during inference to optimize latency. This sort of framework is unfortunately an exception to the rule, owing to Apple's unique control over their entire development stack.

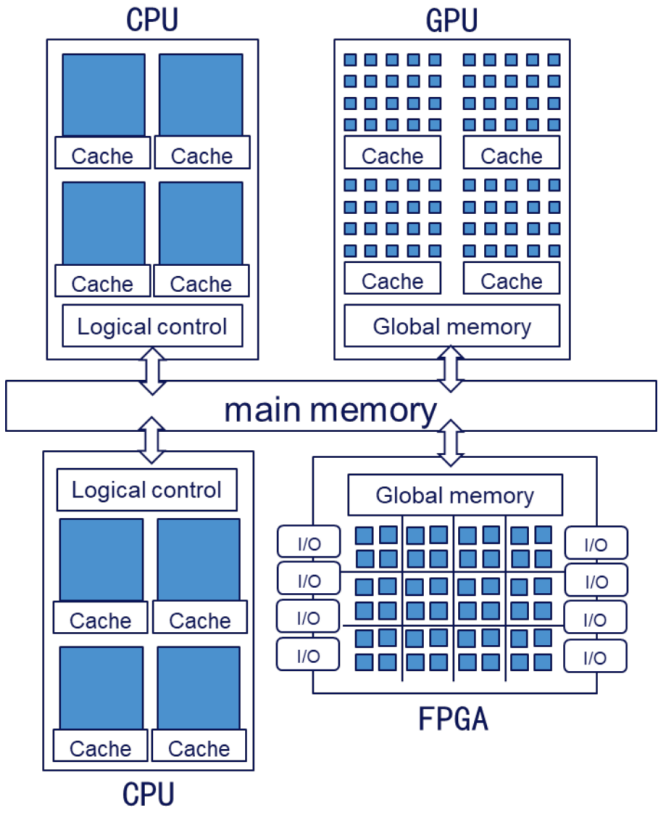

Figure 3: Example of a heterogeneous computing system, featuring a CPU, GPU and FPGA (field-programmable gate array) with unified memory. Adapted from Wu, Q et al (2022) [9].

The Future of Heterogeneous Programming Languages + Edge AI

Heterogeneous computing is a paradigm that combines multiple computing devices with varying architectures into a unified system. It is a unique combination of parallel and distributed computing which uses a central computer to execute SIMD or MIMD instructions. A few existing API's, such as CUDA and OpenCL, have been extended for general heterogeneous programming, and other new API's created. Figure 4 shows a summary of existing languages, distinguishing between novel and extensions to existing languages. There are also many examples of heterogeneous systems in industry. The heterogeneous system architecture (HSA) is a specification developed by the HSA foundation (supported by ARM and AMD) for cross-vendor architectures specifying the design of CPU/GPU integrated platforms. AMD has used this standard in developing its ROCm ecosystem, which uses a single CPU/GPU compiler and for both CPU and GPU instructions [14]. The HSA standard has existed for over ten years, but its widespread adoption in industry appears to be lagging. Developers and engineers argue that the performance benefits of heterogeneous systems are not currently offset by the added complexity and specificity [14]. Significant development is still needed to simplify, unify and extend heterogeneous software to increase general use. Edge AI is an already large industry, poised to undergo rapid growth in the next ten years. Specialized processors and computing architectures, especially heterogeneous systems, will play an important part in this revolution. While heterogeneous architectures have already begun to flood the global market, there is a lack of unified frameworks and programming languages. There will be significant opportunities for enterprises and individuals to play a part in meeting this demand.

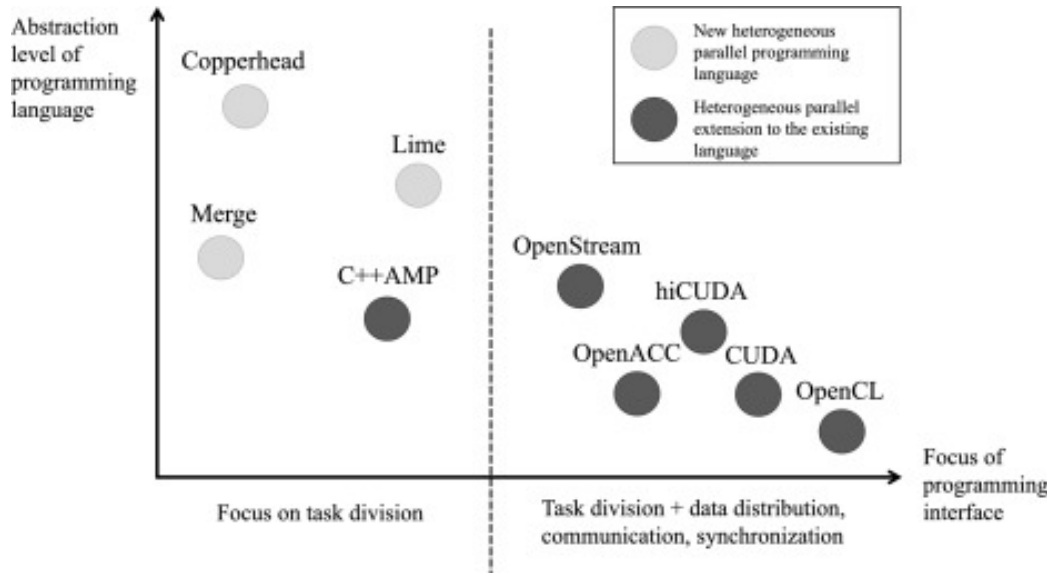

Figure 4: A summary of existing homogenous API's extended and new API's created for heterogeneous computing. Adapted from Wu, Q et al (2022) [13].

References

[1] Singh, R., & Gill, S. S. (2023). Edge AI: a survey. Internet of Things and Cyber-Physical Systems.

[2] Grand View Research (2023). Edge AI Market Size, Share & Trends Analysis Report By Component (Hardware, Software, Edge Cloud Infrastructure, Services), By End-use Industry, By Region, And Segment Forecasts, 2023 - 2030. Market Analysis Report.

[3] Yoo, Hoi-Jun. (2020). Deep Learning Processors for On-Device Intelligence. 1-8. 10.1145/3386263.3409103.

[4] Sipola, T., Alatalo, J., Kokkonen, T., & Rantonen, M. (2022, April). Artificial intelligence in the IoT era: A review of edge AI hardware and software. In 2022 31st Conference of Open Innovations Association (FRUCT) (pp. 320-331). IEEE.

[5] Brodtkorb, A. R., Hagen, T. R., & Sætra, M. L. (2013). Graphics processing unit (GPU) programming strategies and trends in GPU computing. Journal of Parallel and Distributed Computing, 73(1), 4-13.

[6] Jeong, E., Kim, J., Tan, S., Lee, J., & Ha, S. (2021). Deep learning inference parallelization on heterogeneous processors with TensorRT. IEEE Embedded Systems Letters, 14(1), 15-18.

[7] Mittal, S., & Vetter, J. S. (2015). A survey of CPU-GPU heterogeneous computing techniques. ACM Computing Surveys (CSUR), 47(4), 1-35.

[8] Hollemans, M. (2017) A peek inside Core ML. MachineThink. https://machinethink.net/blog/peek-inside-coreml/

[9] Wu, Q., Shen, Y., & Zhang, M. (2022, October). Heterogeneous Computing and Applications in Deep Learning: A Survey. In Proceedings of the 5th International Conference on Computer Science and Software Engineering (pp. 383-387).

[10] Buber, Ebubekir & Diri, Banu. (2018). Performance Analysis and CPU vs GPU Comparison for Deep Learning. 1-6. 10.1109/CEIT.2018.8751930.

[11] Jeong, E., Kim, J., & Ha, S. (2022). Tensorrt-based framework and optimization methodology for deep learning inference on jetson boards. ACM Transactions on Embedded Computing Systems (TECS), 21(5), 1-26.

[12] Karimi, Kamran & Dickson, Neil & Hamze, Firas. (2010). A Performance Comparison of CUDA and OpenCL. Computing Research Repository - CORR. arXiv:1005.2581.

[13] Yunji Chen et al (2024) Chapter 8 - AI programming language for AI computing systems. AI Computing Systems.

[14] Fumero, J. , et al (2019) Programming and Architecture Models in Heterogeneous Computing Architectures. Routledge Handbooks Online.