Analysis of the Designs and Applications of AI Chip

Xinnuo Li

Department of Communications Engineering, Shanghai University, Shanghai, China

bingpohun@shu.edu.cn

Abstract. The rapid evolution of deep learning model architectures and the increasing scale of model parameters have imposed heightened demands on deep learning training, inference, and deployment, leading to the swift advancement and unprecedented prosperity of AI chips. Therefore, this study sets out to analyze the designs and applications of AI chips by considering their unique requirements compared to conventional chips, and by combining software and hardware aspects. The paper delineates the classification of common AI chips along with their distinct design strategies and optimization algorithms. It commences with the fundamental hardware design of AI chips, elucidating the basic design process and addressing the specialized demands of AI computation, particularly data parallelism and storage optimization. Subsequently, transitioning to the manufacturing process, it examines how current AI chips circumvent fabrication bottlenecks and achieve significant breakthroughs in architecture and performance through chip stacking techniques. The paper then bridges hardware and software through the AI compiler, expounding on model optimization approaches, e.g., quantization and pruning, completing the comprehensive journey from AI chip design to deployment. It identifies current developmental challenges in the AI chip realm and provides a glimpse into future prospects. Through a holistic perspective spanning design, manufacturing, algorithms, and applications of AI chips, this paper offers insights that steer upcoming innovations and practical implementations in artificial intelligence, paving the way for a dynamic future in AI chip development.

Keywords: AI chip, Design of AI chip, AI chip structure.

1. Introduction

The development of deep learning algorithms has shown a transformative impact across many fields, encompassing image recognition, speech processing, and natural language understanding. Concurrently, the emergence of generative AI, exemplified by innovations like Stable Diffusion [1], and the ascent of large-scale models such as ChatGPT [2], have not only demonstrated the unprecedented potential of artificial intelligence but have also triggered a paradigm shift in its application. It has propelled industries into a new epoch of holistic intelligence enhancement. However, the pursuit of such advancements necessitates a substantial augmentation in parallel computational capabilities. As the demand for computing power continues to surge and traditional computing architectures prove inadequate to meet the extensive parallel processing demands of deep learning, the software and hardware design of AI chips has garnered increasing attention from numerous companies and researchers. Traditional technology stalwarts in the cloud computing arena, including Google, and IBM, have responded to this trend by introducing specifical AI chip niche. Simultaneously, newer entrants (e.g., Huawei and Cambricon) [3, 4], prominent players in the burgeoning Internet landscape, have strategically positioned themselves by investing in and advancing the AI chip sector. This synchronized effort across the research underscores the urgency and importance of redefining computational infrastructure to harness the full potential of contemporary AI algorithms.

The research directions of AI chips can be categorized into two types: computer science and semiconductor chip technology [5-7] focuses on developing efficient intelligent algorithms, with an emphasis on the implementation, acceleration, and enhancement of network structures. [8-10] focuses on implement and hardware architectures of AI chip design. However, few articles combine both software and hardware aspects. This paper bridges the gap by amalgamating AI chip definition and classification, foundational design principles, design processes, as well as algorithmic and application

considerations. Section 2 will delve into the definition and classification of AI chips, categorizing AI chips into two main types: GPU and ASIC chips, and providing an overview of their general architectures. Section 3 will introduce the design process of AI chips. Starting with a broad perspective encompassing architecture planning, hardware simulation, and physical design aspects, this study will establish a comprehensive understanding of AI chip design. This paper will then proceed to discuss specific optimizations made by AI chip architects for neural networks, focusing on data parallelism and memory hierarchies, as well as highlighting several prominent architectures in use today. Section 4 will delve into the subject of Fabrication. As chip fabrication processes have encountered limitations, chiplets have emerged as a prominent topic. This study will explore the applications of chiplets in the field of AI chips. Section 5 will shift our focus from hardware development to software development. Beginning with compilers, which act as essential bridges between software and hardware, one will explore the specialized optimization techniques employed by AI compilers. As the requirements for performance and computational power differ between the design and deployment phases of neural networks, one will also delve into optimization methods specific to neural network models. Section VI will focus on the limitation and future of AI chips. The lengthy development cycle and high research and development costs of AI chips pose significant challenges in both hardware and software aspects. However, with the advancement of optical chips and AI automation technologies, there is substantial potential for significant improvements in the development process and computational performance of AI chips.

2. Basic Descriptions of AI chip

2.1. Defination

An AI chip pertains to a meticulously engineered chip optimized for the streamlined execution of computations essential for artificial intelligence and machine learning applications, which often involve complex mathematical calculations that traditional central processing units (CPUs) might not be optimized for [9]. Due to the diverse nature of neural network algorithms as well as the varying demands of applications, there exists a wide range of AI chip types. Different chips are tailored to specific professional requirements. AI chips could be generally categorized into two types: general-purpose GPUs and specialized ASICs. General-purpose GPUs (GPGPU) feature a multitude of computing units and extremely long pipelines, exhibiting excellent acceleration effects for computationally intensive tasks in deep learning. On the other hand, specialized ASIC chips are meticulously designed and manufactured based on the specific requirements of certain users and electronic systems. These chips aim at achieving higher processing speeds and lower energy consumption for dedicated tasks, being tailored to deliver optimal performance for the intended task and offers enhanced efficiency compared to general-purpose processors.

2.2. Clasification

2.2.1 GPU

The GPU (Graphics Processing Unit) is a specialized processor originally designed for efficient image and graphics processing. Because of the requirement of storage units and control logic, CPUs excel in logic control but are severely limited in large-scale parallel computing capabilities [11, 12]. By contrast, GPUs feature a higher number of cores and greater memory bandwidth, containing thousands of arithmetic logic units (ALUs) within a single processor, which enables the simultaneous execution of thousands of multiplications and additions. Fig. 1 illustrates the performance advantage of GPUs compared to CPUs. Originally designed to handle three-dimensional graphics, GPUs have evolved from their graphical origins. As computational demands increased and technologies like deep learning and artificial intelligence gained prominence, the computational prowess of GPUs transformed them into crucial tools for achieving efficient computations and processing massive datasets. GPUs also offer a rich array of computational libraries and tools to meet diverse needs across

various scenarios. When applications require flexible precision control or the simultaneous processing of different types of computational tasks, GPUs are an excellent choice. Users can quickly build high-performance neural network models using user-friendly and interactive deep learning frameworks without needing to concern themselves with the underlying hardware implementation and optimization algorithms.

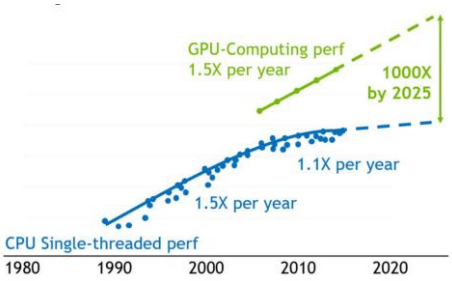

Figure 1. Trend of GPU/CPU performance gap [13]

However, when comparing GPUs to ASICs, the advantages of parallel computing cannot be fully harnessed when applied to deep learning algorithms because of its original design of coping with large-scale image processing. For instance, GPUs are not optimized for memory read and write operations because of its Von Neumann Architecture. Nevertheless, to better cater to deep learning scenarios, graphics card manufacturers have undertaken hardware and software optimizations. Industry giant Nvidia has developed Tensor Cores [14] to enable mixed-precision computation. These cores dynamically adjust computational power based on precision reduction, significantly reducing the time needed from training to convergence while maintaining accuracy. This has led to a substantial improvement in performance and has introduced a range of precisions (TF32, Bfloat16, FP16, FP8, and INT8) that facilitate reliable deployment for inference, maximizing utilization. Nvidia has also developed the CUDA Toolkit [15], integrating it into mainstream deep learning frameworks such as TensorFlow, PyTorch, and MXNet. This integration reduces the entry barriers, expediting the development and deployment of GPU-accelerated applications.

2.2.2 ASIC

ASIC (Application Specific Integrated Circuit) is a type of integrated circuit designed for specific purposes. It cannot be reprogrammed and offers high efficiency with low power consumption. Once manufactured, it cannot be altered, resulting in high initial costs and lengthy development cycles, thereby raising the entry barrier. In recent years, various chips such as TPUs, NPUs, VPUs, and BPUs have emerged, all of which fundamentally belong to the category of ASICs. Due to their perfect suitability for neural network-related algorithms, ASICs outperform GPUs in terms of performance and power consumption. Currently, most ASIC chips involve giants with expertise in both AI algorithms and chip development. For instance, the Tensor Processing Unit (TPU) is a specialized ASIC developed by Google [16]. It serves as a matrix processor specifically designed for neural network workloads. TPUs employ hardware tailored for the execution of large-scale matrix operations commonly found in machine learning algorithms. They also feature High Bandwidth Memory (HBM), enabling efficient training and accommodating larger models. TPUs are comprised of numerous multiplication-accumulation units meticulously crafted for matrix computations. Through a systematic systolic array architecture, these units are intricately interconnected to form a large physical matrix. Every multiplication executed passes its results into the subsequent multiplier. This process occurs without the need for memory access during computation, thereby enabling TPUs to attain unparalleled computational efficiency in neural network calculations [17].

Researchers have also made breakthroughs in attempts to depart from the von Neumann architecture, introducing the Neural Network Processing Unit (NPU). In the von Neumann architecture, storage and processing are separated and accomplished by memory and arithmetic units, respectively, which inevitably affects efficiency when running neural network applications. In neural networks, however, storage and processing are integrated, both being represented through synaptic

weights. The working principle of an NPU involves circuitry that simulates human neurons and synapses, and it directly processes large-scale neurons and synapses via a specialized deep learning instruction set. Each instruction performs processing for a group of neurons. Compared to CPUs and GPUs, NPUs achieve storage and computation integration through synaptic weights, amplifying their operational efficiency. Currently, numerous manufacturers have introduced their own NPU products, including Huawei's Ascend NPU and Samsung's Neural Processing Unit.

3. Design and principle

3.1. Overall Process

Designing an AI chip is a complex and multidisciplinary process that involves various stages, including architectural design, hardware implementation, software development, and testing. It shares several commonalities with general chip design [18]. The initial step involves defining the system architecture and devising the algorithms that will govern the chip's behavior, determining how various hardware components will interact to execute complex AI tasks efficiently. In this process, Unified Modeling Language (UML) is also frequently used to visually communicate the system's structure, components, and their interconnections. Once the architecture is chosen, the next step involves system-level simulation to ensure that the designed system meets the Key Performance Metrics (KPM) set in the product planning. High-level languages such as Python, C, C++, are employed for architectural validation from functional models. Subsequently, property specification languages (such as PSL and SVA) are used to define system requirements and create mathematical models of the realized system. Next, Hardware Description Languages (HDL) like VHDL and Verilog are employed to capture the entire system design process, including detailed information about I2C input and output pins, IP block instances, design connections, clock, and reset strategies. This stage involves initial logic functionality verification and does not incorporate the simulation of parasitic distributed parameters. After simulation, design specifications are specified. Aspects like power consumption, processing speed, memory capacity, and precision requirements need to be considered. These specifications serve as guiding principles throughout the design and development process. Finally, one comes the physical implementation. Engineers use Electronic Design Automation (EDA) software tools to translate the high-level system architecture into detailed circuit layouts and use EDA tools to simulate the performance of the designed circuits. After the design part of the chip concludes, the physical layout is provided to chip foundries to manufacture the actual circuit on silicon wafers, followed by packaging and testing, resulting in the physical chip one sees in reality.

3.2. Specified Design Process

In terms of architectural design, due to the sensitivity of neural networks to both data parallelism and data storage and access, ASIC chips related to neural networks undergo specialized optimizations in these two aspects, distinguishing them from CPUs and ASICs in other domains.

3.2.1 Data parallelism

While both AI and general chips may incorporate parallelism, AI chips often emphasize massive parallelism to handle the concurrent processing demands of neural networks, placing specific emphasis on architectural features that optimize for AI workloads, such as matrix multiplications, convolutions, and neural network operations. The AI chip design involves two distinct architectural paradigms: temporal computing architecture and spatial computing architecture. Temporal computing involves utilizing specialized processors with custom instruction sets. These processors manage and schedule logic computation units (Arithmetic Logic Units, ALUs) and storage systems through instruction pipelines. Given that deep learning neural network models mainly consist of numerous linear algebra operations with relatively simple control flows, AI chips often employ highly parallel computing architectures. This design choice enhances processor parallelism by performing the same operation on multiple data elements simultaneously. Based on the fixed computational workflow of

deep learning, multiple ALU pipelines can be designed to efficiently execute fixed-step calculations, such as the multi-data multiplication and accumulation calculations in matrix operations. Notably, the DianNao architecture developed by the Institute of Computing Technology, Chinese Academy of Sciences, is a typical example of temporal computing architecture (Fig. 2) [19]. The architecture comprises Neural Functional Units (NFUs), multiple functional on-chip storage areas, and controllers. The NFUs and storage are coordinated through controller instructions. The NFU operation core is divided into three stages: NFU1, responsible for parallel multiplication operations; NFU2, housing addition trees; and NFU3, implementing nonlinear activation functions. NFU-1 features a 16x16 parallel multiplication unit capable of executing 256 multiplication operations in a single cycle. NFU-2 contains 16 addition trees, each consisting of 15 adders arranged in an 8-4-2-1 structure. NFU-3 includes 16 activation function units, which promotes parallelism and efficient computation.

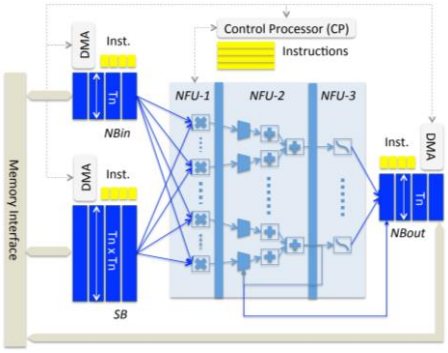

Figure 2. DianNao Architecture

In contrast, spatial computing architecture employs individual processing elements (PEs), each equipped with its own control and storage units. Spatial computing arrays typically consist of one-dimensional or two-dimensional PE arrays, where each PE comes with an embedded controller and cache. PEs can directly exchange data between each other. The architecture is complemented by on-chip global caches, off-chip DRAM, and multiple-level storage systems. This setup exploits a multitude of PE arrays to achieve efficient parallel computation, while data movement between PEs reduces the need for frequent communication between processors and main memory. Google's Tensor Processing Unit (TPU) exemplifies spatial computing architecture (seen from Fig. 3) [20]. TPU features a 256x256 matrix multiplication array composed of MAC units, forming a two-dimensional array. This architecture leverages a systolic array interconnection method, where data for PE computations originates from neighboring PEs in the previous clock cycle. The resulting computations flow to adjacent PEs in the subsequent clock cycle, resembling the pulsating blood flow in vessels. This mechanism is referred to as the systolic array architecture. In operation, instructions and data enter the TPU through the host interface. Preloaded weight parameters with higher reuse rates are stored in a Weight FIFO, and input data resides in a unified buffer (UB). The matrix multiplication occurs within PE units, and the results pass to the accumulation unit (Acc). Depending on the model's design requirements, the accumulated results may undergo Activation, Normalization, and Pooling operations. Finally, the results are sent back to the unified buffer.

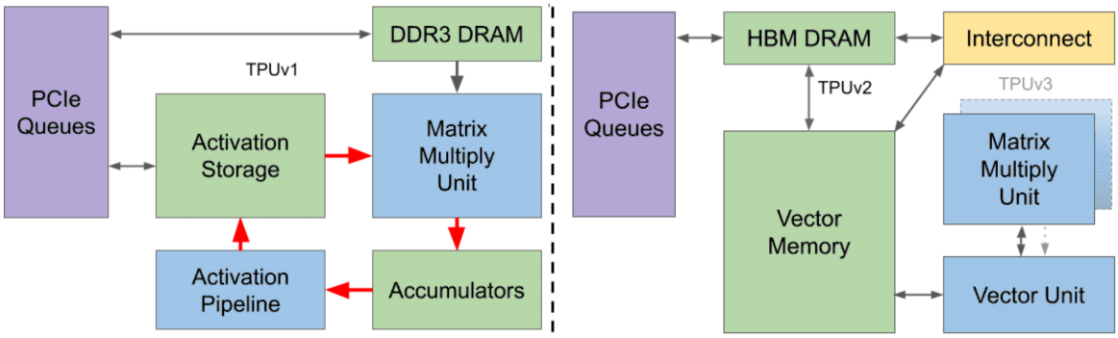

Figure 3. TPU block diagram

3.2.2 Specialized memory hierarchies

As parallelism increases, the reading and writing of data can become a bottleneck, leading to what is commonly referred to as the "memory wall." Deep neural networks consist of numerous network layers, each containing a substantial number of parameters and intermediate feature data. Both data access and computational requirements are extensive. For instance, convolutional layers involve a multitude of multidimensional convolution kernels. The sliding window nature of convolution operations leads to repeated participation of convolution kernel parameters in computations. Given that the number of convolution kernel parameters often far exceeds the cache capacity, parameters need to be accessed repeatedly from the main memory. For example, early models like AlexNet [21] contained 60 million parameters, while recent architectures based on the Transformer [22] framework, possess a staggering 175 billion parameters. Such immense parameter volumes not only consume significant storage space but also impose stringent demands on memory bandwidth, memory management, and computation efficiency. Due to the fact that the computation results of the previous layer in a deep learning model are utilized in subsequent layers, model parameters are reused across different clock cycles. Memory access challenges can be addressed through various strategies: increasing on-chip cache sizes and the number of registers, boosting memory bus bandwidth to enhance memory access efficiency, and subsequently reducing data waiting times.

In recent years, to overcome the bottleneck of memory access in traditional von Neumann architectures, the integration of storage and computation into a unified architecture has emerged as a significant trend in AI chip development. The concept of storage-computation integration involves embedding computational capability within memory, employing novel computational architectures for two-dimensional and three-dimensional matrix multiplication and addition operations. This approach expands the design space of chips, making them well-suited for a wide range of intelligent computing applications. For instance, the storage-computation integrated chip developed by Alibaba DAMO Academy incorporates near-memory computing using heterogeneous integrated embedded SeDRAM [23]. In SeDRAM-based storage-computation chips, AI circuits and peripheral circuits including control, I/O, and DFT (Design for Testability) are segregated into a logic chip, stacked atop the memory array chip using hybrid Cu-to-Cu bonding. The DFT module is designed as IP within the logic chip, serving the array chip for Built-In Self-Repair (BISR) operations.

In addition to near-memory computing, the concept of storage-computation integration is also evolving towards in-memory computing. Notable examples include companies like Mythic, Witmem and Reexen. In this direction, computational operations are executed by independent computing units within the storage chip/region. Storage and computation can be either analog or digital. This approach is suitable for scenarios where the algorithms are fixed and computations are relatively lightweight, such as in speech processing.

4. Fabrication

After the design phase, the manufacturing process is carried out by wafer fabrication facilities. This involves steps such as applying photoresist, UV exposure, and etching to create the chip's components. The fundamental building blocks of a chip are transistors, and a large-scale chip can contain billions of transistors. Transistors contain a gate, and the width of this gate represents the chip's nanometer manufacturing process. A narrower gate results in lower heat generation and power consumption, allowing more transistors on the chip and enhancing performance and computational capacity. As chip computational power grows exponentially, chip sizes have started exceeding the dimensions of photolithography mask templates, causing process upgrades to face bottlenecks. Chiplets and heterogeneous integration technologies have emerged as effective approaches to maintain Moore's Law and overcome template limitations.

Chiplets are a silicon-level reuse technology that offers customized solutions in a rapid and cost-effective manner. For instance, different applications might require varying computational capabilities while using the same core, memory, and I/O components. Chiplet technology enables

packaging the most suitable chiplet for a specific function. This allows packaging memory, logic, analog, and co-packaged optical devices, each with different manufacturing processes. Furthermore, chiplets address data flow issues. In cloud-based AI acceleration, the interconnection between the host CPU and AI acceleration chip, as well as between multiple acceleration chips, currently relies on technologies like PCIe and NvLink [24]. Utilizing silicon-level interconnects like chiplets can significantly improve bandwidth, latency, and power consumption.

To realize this novel IP reuse paradigm of chiplets, advanced chip integration and packaging technologies are essential. Technologies like Intel's Foveros and 3D integration [25] convert multi-chip packaging from a single plane to a three-dimensional composition, greatly enhancing integration density and enabling flexible combinations of chips or functional modules. Similar concepts are reflected in Tesla's Dojo chip [26], with each D1 training module composed of a 5x5 array of D1 chips interconnected in a two-dimensional mesh structure and employing InFO_SoW (Silicon on Wafer) packaging to enhance chip-to-chip interconnect density. In addition to proprietary interconnect standards and integrated packaging technologies developed by industry giants, open standards like UCIe (Fig. 4) [27] have emerged. These open standards facilitate seamless interoperability between chiplets from different manufacturers, potentially becoming the pervasive packaging interconnect for small chiplets and driving the growth of an open small chip ecosystem. Furthermore, the chiplet model introduces a new option beyond traditional IP and chip suppliers: chiplet silicon providers. For current AI chip manufacturers, they can either focus on AI acceleration and offer products in the form of IP or external hardware accelerator chips, or they can adopt a vertical approach and create SoCs integrating AI acceleration functionalities. Chiplets offer a new product format for the former, expanding potential markets or extending the lifecycle of a generation of products (technologies). For companies with strong silicon implementation capabilities, specializing in chiplets could be a future direction. For the latter, direct integration of suitable AI chiplets instead of IP (requiring their own chip implementation) can significantly save project development time. It is foreseeable that AI chiplets will become a crucial mode for AI hardware reuse and integration.

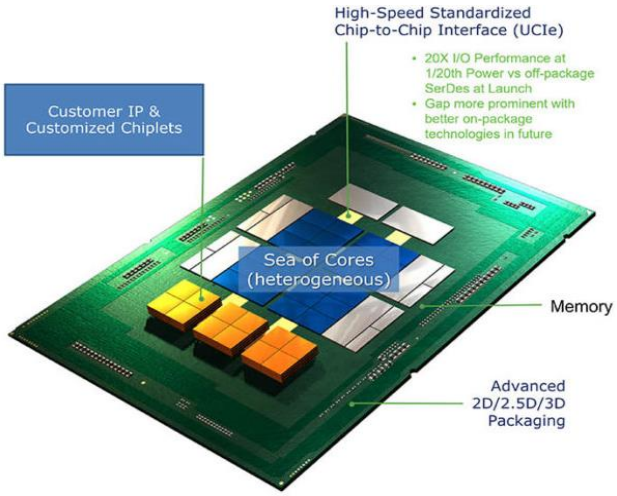

Figure 4. Universal Chiplet Interconnect Express UCIe 1.0 Cover

5. Application and Algorithms

When deploying deep learning models to edge computing devices or for commercial applications, two main aspects need to be considered. On one hand, the computational graph and operators within the model need to be translated into lower-level machine language for faster execution. This requires the use of AI compilers specialized for optimizing neural networks. On the other hand, the network structure of the model itself needs optimization to enhance computational speed and required computational resources, while minimizing performance loss as much as possible [28].

5.1. AI compiler

As an integral piece of supporting software for chips, compilers play a crucial role in enhancing the development efficiency and compatibility of application software by abstracting high-level languages. They also adapt to the underlying architecture to generate efficient executable code, serving as a bridge between software and hardware. As the core abstraction in AI compilers is tensors, one optimization direction of AI compilers focuses on tensors. In the field of AI, the processing of various data types such as images, text, and videos are abstracted into tensor operations, such as convolutions, transpositions, pooling, normalization, etc. These tensor operations form a dataflow graph representing tensor computations when sequenced. Since traditional compilers often focus on scalar to vector-level data processing granularity, prioritizing universality as a key starting point in designing compilation optimization, which makes them ill-equipped for analyzing low-level Intermediate Representations (IR). AI compilers typically search through scheduling spaces to find optimal tensor optimization strategies that fit the backend.

For instance, TVM is an example in this context. Figure 5 illustrates the workflow of TVM [29]. It begins by reading models from existing frameworks, generating a computational graph representation. Then, high-level dataflow rewriting takes place, producing an optimized computational graph. Subsequently, operator-level optimization occurs to generate efficient code for fused operators within the computational graph. Following this, utilizing an ML optimizer model, the algorithm searches for the optimal code from a potential optimization set on the target machine, encapsulating the generated code into deployable modules. It offers a unified IR stack and automated generation techniques. By automatically transforming the computational graph, it achieves fusion of computation patterns, efficient memory utilization, and optimization of data layout, completing the optimization from the graph level to the operator level.

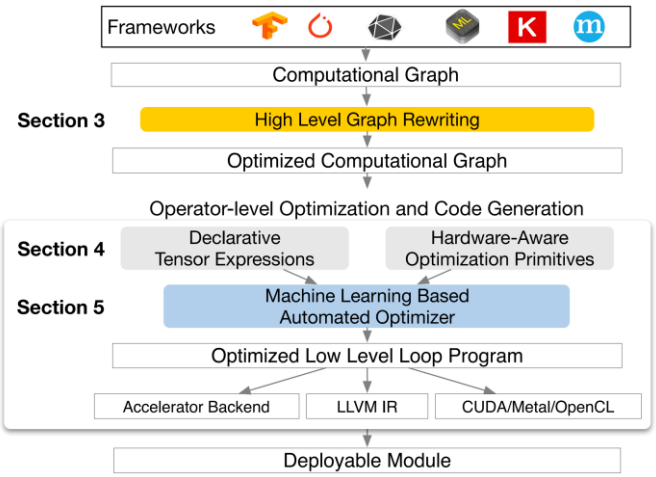

Figure 5. System overview of TVM

Another optimization direction involves interfacing with dynamic interpreter languages primarily based on Python. It is a challenge for AI compilers to convert it to a static IR poses for Python is an extremely flexible language for interpreted execution. For example, traditional PyTorch programming cannot easily extract the model's graph structure from Python code. This limitation hinders tasks such as model exporting, operator fusion optimization, and model quantization. Addressing this challenge, a notable advancement is the JIT compilation interface called TorchDynamo, proposed in PyTorch 2.0 [30]. TorchDynamo supports modifying Python's dynamic execution logic at runtime, just before CPython interpreter bytecode execution. This allows setting a custom frame at runtime, and the bytecode within this frame can be called back to the Python layer for modification, enabling users to customize the computation graph. Utilizing this mechanism, TorchDynamo supports the characteristics of dynamic graphs, automatically invoking corresponding static sub-models through Python's execution mechanism. In the context of Automatic Mixed Precision, the model's training speed was increased by 43% [30]. Additionally, there's a focus on

global optimization for neural networks. Specific optimizations introduced by AI compilers include tensor optimization, automatic differentiation, and automatic parallelism. Tensor optimization focuses on the abstraction of computation into tensor operations, and its optimization is pivotal in AI compiler design. Automatic differentiation, crucial for AI network training, supports gradient descent and error backpropagation, necessitating automatic differentiation technology. Automatic parallelism techniques include data parallelism, operator-level model parallelism, pipeline model parallelism, optimizer model parallelism, and recomputation. These techniques aim to enhance overall parallelism to optimize network performance in the AI domain.

5.2. Model optimization

When designing DNN models, the primary goal is often to improve accuracy without significant consideration for implementation complexity. This approach can lead to DNN networks that are challenging to implement or deploy. One characteristic of deep learning is that it doesn't require very high computational precision [31]. Reducing the bit-width of CNN weights can substantially decrease storage requirements and computational complexity. For instance, DNN inference has transitioned from initially requiring 32-bit operations to gradually using 16-bit and 8-bit, or even fewer, operations. The essence of this issue lies in quantizing data into a smaller set of quantization levels, with the ultimate goal of minimizing quantization error, the difference between quantized and original data. Specific quantization methods include linear and nonlinear quantization. For example, Binary Connect (BC) uses two weights, -1 and +1, replacing multiplication with addition. Recently, Ternary Weight Nets (TWN) [32] emerged, adding a "0" to binary weights. It achieved a modest 3.7% accuracy loss compared to 32-bit floating-point numbers. Piecewise Linear Quantization (PWLQ) further reduced Inception-v3's accuracy loss to 1.04% [33].

Nonlinear quantization techniques encompass log-domain quantization and weight sharing. The former involves quantization of data that follows a logarithmic distribution, or employing this approach for nonlinear quantization. Weight sharing forces multiple weights to share the same value, significantly reducing the total number of unique weight values. This method requires storing both the weight value and an index indicating which weight value to use at a particular position within a filter or kernel. To read weights, one must first read the weight index and then fetch the corresponding weight value. The article summarizes the achieved bit-width and corresponding accuracy loss using these methods. Another direction involves reducing the number of operations and the size of DNN models. If the same or similar accuracy can be achieved with fewer operations or a smaller network, implementation complexity can be significantly reduced. This work can be broadly categorized into four classes: (1) Network pruning: Pruning techniques can remove 90% of parameters while maintaining accuracy unchanged [34]. It involves removing unnecessary connections or neurons from the network to reduce the model's complexity; (2) Utilizing sparsity: Certain typical model architectures and operations, such as ReLU and dropout, introduce a large number of zero values into activations, which can reduce storage and memory bandwidth requirements. Skipping zero values can also decrease MAC operations. Additionally, carefully treating values very close to "0" as "0" can further enhance sparsity; (3) Compact network architectures: Designing smaller network structures that maintain or approximate the same level of accuracy as larger networks. Fig. 6 illustrates that as neural networks evolve, models with fewer parameters and improved performance are gradually being developed; (4) Knowledge distillation: This technique transfers knowledge from a complex model (teacher) to a simpler model (student). By doing so, the student network can achieve accuracy that would otherwise be challenging to attain directly using the same dataset for training.

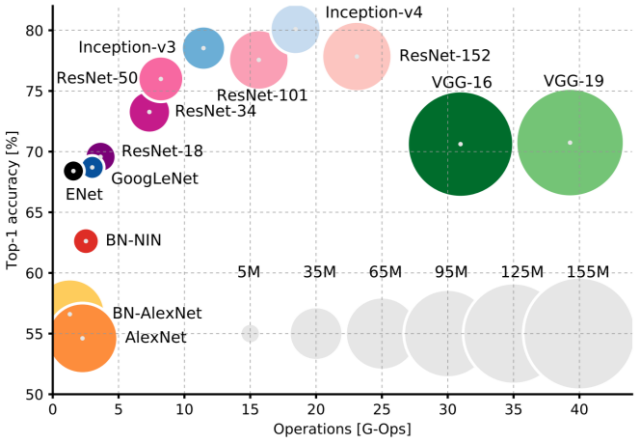

Figure 6. A scatter plot of network sizes, performances of DNN [35]

6. Limitations & Prospects

Although AI chips, like CPUs, are constrained by fabrication processes, specialized optimizations in both hardware and software through research have granted AI chips computational capabilities far surpassing those of CPUs. However, the production process of AI chips is lengthy and involves collaborative operations between hardware and software. On the hardware side, emerging data parallelism and storage technologies, such as integrated storage-computation designs and three-dimensional stacking, are influenced by factors like technology, cost, and quality control. These technologies require substantial financial and technical resources, and there is still a considerable gap between their development and practical implementation. Additionally, relying solely on specialized architectures no longer carries a high entry barrier, making differentiation increasingly challenging.

On the software side, attention is now focused on the chip's accompanying toolchain. Furthermore, software tool optimizations are virtually limitless – addressing single-core challenges leads to multi-core, multi-chip, and multi-board challenges. Model updates and iterations occur rapidly, demanding continuous efforts. Transforming chip products into real-world applications presents even greater challenges than the chip development process itself, requiring significant software development and customer support efforts.

Nevertheless, intense competition has spurred the emergence of numerous nascent technologies that hold the potential to significantly enhance AI chip performance. For example, the MIT Optical AI team has applied "time-space multiplexing" to optical chip design concepts and employed optoelectronics for multiplication and addition operations [36]. Due to the ultra-high optical bandwidth and low-loss data transmission, optical computing chips could achieve several orders of magnitude in computational breakthroughs in the short term, becoming the next-generation computing processors.

Furthermore, as AI-designed chips continue to evolve, more automated design tools can play a vital role in propelling the development of the AI chip industry. For instance, Synopsys has introduced a comprehensive AI-driven EDA solution, covering advanced digital and analog chip design, verification, testing, and manufacturing. The Chinese Academy of Sciences has used AI technology to design the world's first CPU chip that is entirely self-generated and devoid of manual intervention [37], based on the Binary Speculation Diagram algorithm. Envisioning a future where AI designs AI, this direction is poised to shape the development of AI chips.

7. Conclusion

This paper comprehensively delves into the designs and applications of AI chips, marking a paradigm shift in the evolution of artificial intelligence. While there are numerous architectures and types, AI chips can be classified into two categories: general-purpose GPUs and ASIC chips. Through

specialized mixed-precision calculations and dataflow architectures, they have significantly outperformed CPUs in various ways. In terms of design and manufacturing, AI chips share similarities with common chips, but they differ significantly in terms of data parallelism and storage structures. Designs such as spatial-temporal parallelism or integrated storage-computation address process-related bottlenecks. Moreover, methods like 3D stacking integrate components of different processes onto a single chip, increasing bandwidth and reducing costs.

On the software horizon, the compiler functions as a crucial bridge between the software and hardware aspects, exerting a significant influence on the performance of the model. Besides, model optimization underscores an essential role in the practical realization of AI chips, where achieving optimal performance relies upon a strategic interplay between software and hardware elements. Intense competition within the AI chip arena, coupled with factors like technology, cost, and quality control, necessitates substantial financial and technical investments, driving significant costs from research to deployment. However, these challenges also fuel the flourishing development of new technologies. Optical computing and the advancement of AI-designed chips underline the potential for transformative breakthroughs, being expected to play a significant role in improving performance and reducing costs. In conclusion, by incorporating a holistic view that spans AI chip definition, classification, design principles, processes, algorithms, and applications, this paper illuminates the intricate terrain of AI chip evolution. It offers insights that guide forthcoming innovations and practical implementations within artificial intelligence, paving the way for a dynamic future in AI chip development.

References

[1] Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10684-10695. Fangfang. Research on power load forecasting based on Improved BP neural network. Harbin Institute of Technology, 2011.

[2] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. Advances in neural information processing systems, 2020, 33: 1877-1901.

[3] Huawei Technologies Co., Ltd. Huawei MindSpore AI Development Framework. Artificial Intelligence Technology. Singapore: Springer Nature Singapore, 2022: 137-162.

[4] Zhao Y, Liu C, Du Z, et al. Cambricon-Q: A hybrid architecture for efficient training. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2021: 706-719.

[5] Lavin A, Gray S. Fast algorithms for convolutional neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 4013-4021.

[6] Sze V, Chen Y H, Yang T J, et al. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 2017, 105 (12): 2295-2329.

[7] Mittal S. A survey on modeling and improving reliability of DNN algorithms and accelerators. Journal of Systems Architecture, 2020, 104: 101689.

[8] Dean J, Patterson D, Young C. A new golden age in computer architecture: Empowering the machine-learning revolution. IEEE Micro, 2018, 38 (2): 21-29.

[9] Li B, Gu J, Jiang W. Artificial intelligence (AI) chip technology review. 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI). IEEE, 2019: 114-117.

[10] Yan G, Lu W, Li X, et al. Comparative study of the domain-specific processors. Scientia Sinica Informationis, 2022, 52 (2): 358-375.

[11] Lee V W, Kim C, Chhugani J, et al. Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. Proceedings of the 37th annual international symposium on Computer architecture. 2010: 451-460.

[12] Li B, Gu J, Jiang W. Artificial intelligence (AI) chip technology review. 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI). IEEE, 2019: 114-117.

[13] Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. New plot and data collected for 2010-2015 by K. Rupp.

[14] Markidis S, Der Chien S W, Laure E, et al. Nvidia tensor core programmability, performance & precision. 2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW). IEEE, 2018: 522-531.

[15] Ilievski A, Zdraveski V, Gusev M. How CUDA powers the machine learning revolution. 2018 26th Telecommunications Forum (TELFOR). IEEE, 2018: 420-425.

[16] Jouppi N, Kurian G, Li S, et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. Proceedings of the 50th Annual International Symposium on Computer Architecture. 2023: 1-14.

[17] Google cloud. Retrieved from: https://cloud.google.com/tpu/docs/system-architecture-tpu-vm

[18] Martin G, Chang H. System-on-Chip design. ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No. 01TH8549). IEEE, 2001: 12-17.

[19] Chen T, Du Z, Sun N, et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGARCH Computer Architecture News, 2014, 42 (1): 269-284.

[20] Jouppi N P, Yoon D H, Ashcraft M, et al. Ten lessons from three generations shaped google's tpuv4i: Industrial product. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2021: 1-14.

[21] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012, 25.

[22] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems, 2017, 30.

[23] Niu D, Li S, Wang Y, et al. 184QPS/W 64Mb/mm 2 3D logic-to-DRAM hybrid bonding with process-near-memory engine for recommendation system. 2022 IEEE International Solid-State Circuits Conference (ISSCC). IEEE, 2022, 65: 1-3.

[24] Temuçin Y H, Sojoodi A H, Alizadeh P, et al. Efficient multi-path NVLink/PCIe-aware UCX based collective communication for deep learning. 2021 IEEE Symposium on High-Performance Interconnects (HOTI). IEEE, 2021: 25-34.

[25] Ingerly D B, Amin S, Aryasomayajula L, et al. Foveros: 3D integration and the use of face-to-face chip stacking for logic devices. 2019 IEEE International Electron Devices Meeting (IEDM). IEEE, 2019: 19.6. 1-19.6. 4.

[26] Talpes E, Williams D, Sarma D D. Dojo: The microarchitecture of tesla's exa-scale computer. 2022 IEEE Hot Chips 34 Symposium (HCS). IEEE Computer Society, 2022: 1-28.

[27] Sharma D D, Pasdast G, Qian Z, et al. Universal chiplet interconnect express (UCIe): An open industry standard for innovations with chiplets at package level. IEEE Transactions on Components, Packaging and Manufacturing Technology, 2022, 12 (9): 1423-1431.

[28] Universal Chiplet Interconnect Express UCIe 1.0 Launched. Retrieved from: https://www.servethehome.com/universal-chiplet-interconnect-express-ucie-1-0-launched/.

[29] Chen T, Moreau T, Jiang Z, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), 2018: 578-594.

[30] PYTORCH ORGANIZATION. 2022. API Index. Retrieved from: https://pytorch.org/tutorials/intermediate/dynamo_tutorial.html.

[31] Micikevicius P, Narang S, Alben J, et al. Mixed precision training. arXiv preprint arXiv: 1710.03740, 2017.

[32] Liu B, Li F, Wang X, et al. Ternary weight networks. ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023: 1-5.

[33] Fang J, Shafiee A, Abdel-Aziz H, et al. Post-training piecewise linear quantization for deep neural networks. Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. Springer International Publishing, 2020: 69-86.

[34] Frankle J, Carbin M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv: 1803.03635, 2018.

[35] Canziani A, Paszke A, Culurciello E. An analysis of deep neural network models for practical applications. arXiv preprint arXiv: 1605.07678, 2016.

[36] Chen Z, Sludds A, Davis III R, et al. Deep learning with coherent VCSEL neural networks. Nature Photonics, 2023: 1-8.

[37] Cheng S, Jin P, Guo Q, et al. Pushing the Limits of Machine Design: Automated CPU Design with AI. arXiv preprint arXiv: 2306.12456, 2023.