EQ-ViT: Algorithm-Hardware Co-Design for End-to-End Acceleration of Real-Time Vision Transformer Inference on Versal ACAP Architecture

Peiyan Dong, *Graduate Student Member, IEEE*, Jinming Zhuang, *Graduate Student Member, IEEE*, Zhuoping Yang, Shixin Ji, Yanyu Li, Dongkuan Xu, *Member, IEEE*, Heng Huang, Jingtong Hu, *Senior Member, IEEE*, Alex K. Jones, *Fellow, IEEE*, Yiyu Shi, *Senior Member, IEEE*, Yanzhi Wang, *Senior Member, IEEE*, and Peipei Zhou, *Senior Member, IEEE*

***Abstract*—While vision transformers (ViTs) have shown consistent progress in computer vision, deploying them for real-time decision-making scenarios (< 1 ms) is challenging. Current computing platforms like CPUs, GPUs, or FPGA-based solutions struggle to meet this deterministic low-latency real-time requirement, even with quantized ViT models. Some approaches use pruning or sparsity to reduce the model size and latency, but this often results in accuracy loss. To address the aforementioned constraints, in this work, we propose EQ-ViT, an end-to-end acceleration framework with the novel algorithm and architecture co-design features to enable the real-time ViT acceleration on the AMD Versal adaptive compute acceleration platform (ACAP). The contributions are four-fold. First, we perform in-depth kernel-level performance profiling and analysis and explain the bottlenecks for the existing acceleration solutions on GPU, FPGA, and ACAP. Second, on the hardware level, we introduce a new spatial and heterogeneous accelerator architecture, the EQ-ViT architecture. This architecture leverages the heterogeneous features of ACAP, where both FPGA and artificial intelligence engines (AIEs) coexist on the same system-on-chip (SoC). Third, On the algorithm level, we create a comprehensive quantization-aware training strategy, the EQ-ViT algorithm. This strategy concurrently quantizes both the weights and activations into 8-bit integers, aiming to improve the accuracy rather than compromise it during quantization. Notably, the method also quantizes nonlinear functions for efficient hardware implementation. Fourth, we design the EQ-ViT automation framework to implement the EQ-ViT architecture for four different ViT applications on the AMD Versal ACAP VCK190 board, achieving accuracy improvement with 2.4%, and average speedups of 315.0, 3.39, 3.38, 14.92, 59.5, and 13.1× over computing solutions of Intel Xeon 8375C vCPU, Nvidia A10G, A100, Jetson AGX Orin GPUs, AMD ZCU102, and U250 FPGAs. The energy efficiency gains are 62.2, 15.33, 12.82, 13.31, 13.5, and 21.9×.**

***Index Terms*—Design for space exploration, embedded systems, FPGA, hardware/software co-design, high-level synthesis, modeling, performance optimization, reconfigurable logic.**

Manuscript received 9 August 2024; accepted 10 August 2024. This work was supported in part by the NSF under Award 2213701, Award 2217003, Award 2133267, Award 2122320, Award 2324864, Award 2324937, and Award 2328972; and in part by the NIH under Award R01EB033387. This article was presented at the International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS) 2024 and appeared as part of the ESWEEK-TCAD Special Issue. This article was recommended by Associate Editor S. Dailey. *(Peiyan Dong and Jinming Zhuang are co-first authors.) (Corresponding author: Peipei Zhou.)*

Peiyan Dong, Yanyu Li, and Yanzhi Wang are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: dong.pe@northeastern.edu; li.yanyu@northeastern.edu; yanz.wang@northeastern.edu).

Jinming Zhuang, Zhuoping Yang, Shixin Ji, and Peipei Zhou are with the School of Engineering, Brown University, Providence, RI 02912 USA (e-mail: jinming_zhuang@brown.edu; zhuoping_yang@brown.edu; shixin_ji@brown.edu; peipei_zhou@brown.edu).

Dongkuan Xu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (e-mail: dxu27@ncsu.edu).

Heng Huang is with the Department of Computer Science, University of Maryland, College Park, MD 20742 USA (e-mail: heng@umd.edu).

Jingtong Hu is with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: jthu@pitt.edu).

Alex K. Jones is with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: akj@syr.edu).

Yiyu Shi is with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: yshi4@nd.edu).

Digital Object Identifier 10.1109/TCAD.2024.3443692

I. Introduction

VISION transformers (ViTs) [1], [2], [3] have shown remarkable versatility in a broad range of application domains, including computer vision (e.g., image classification [1], [3], object detection [4], [5], image processing [6], and video understanding [7]), and in complex scenarios that involve the multimodal data. Many networks [1], [8], [9], [10] use ViTs as the backbone [8], [9] and show superior transferability to various downstream tasks with minor fine tuning.

Low-Latency Real-Time Application Scenarios: Adopting ViT inference as a key chain for low-latency real-time decision making usually requires stringent latency requirements. For example, in autonomous driving scenarios with a 120 km/h speed, 1 ms latency corresponds to 3 cm between a vehicle and a static object or 6 cm between the two moving vehicles [11]. In such a life-critical system, deterministic low latency (<1 ms) is the first-class design citizen. European Organization for Nuclear Research (CERN) collaborates with autonomous driving software company Zenseact to apply CERNâ$^{\text{TM}}$s decision-making algorithm acceleration on FPGA at microsecond level to help avoid accidents in self-driving cars [12]. Such latency (<1 ms) is required in broader scenarios, including the edge and cloud applications. *On the edge*, for example, radio access networks (RANs) [13] support

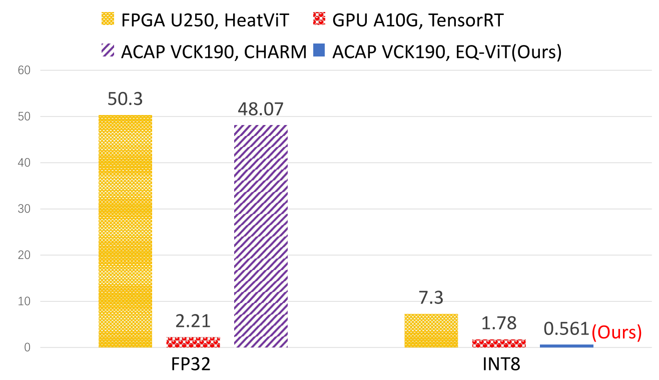

Fig. 1. E2E latency comparison for DeiT-T (FP32, INT8, batch size = 6) by using HeatViT on U250 FPGA, TensorRT on A10G GPU, CHARM on Versal ACAP VCK190, and EQ-ViT (ours) on ACAP VCK190.

interactive streaming media [14], augmented reality/virtual reality (AR/VR) [15], [16], robot systems control [17], online error detection in the manufacturing industry [18], and industrial IoT 4.0 [19]. RAN stack operates in low-latency at a transmission time interval of 1 ms or less (based on the 5G standards). Thus, it has to make control decisions at each millisecond [13]. In AR/VR, the latency requirement is <1 ms as the visual reaction time for the human expected events is only around 1 ms [20]. *In the cloud*, to guarantee the quality of service, deep learning-based inference for the cloud services in Microsoft Bing Search [21], Microsoft Azure Cloud [22], [23], and Google Cloud [24], [25], [26], all have a single-digit millisecond latency budget to process. Powered by the next-generation cellular networks with 5G or 6G standard [13], optical interconnection network [27], and optical chiplet [28], [29] technology, the latency requirement will be more stringent. Acceleration solutions that meet certain end-to-end (E2E) inference latency requirements and optimize the overall system energy efficiency, i.e., performance per watt are desired.

However, the existing works fail to fulfill such stringent low-latency requirements, hindering the ViT deployment in low-latency application scenarios. We measure the E2E low batch inference latency for the representative ViT model DeiT-T [2] using the state-of-the-art (SOTA) acceleration frameworks on the FPGA and GPU, including HeatViT [30] on AMD U250 FPGA, and TensorRT [31] on Nvidia A10G GPU. As shown in Fig. 1, in terms of E2E inference latency under single-precision floating-point (FP32) precision, U250 FPGA takes 50.3 ms, which far exceeds the low-latency real-time requirement, e.g., <1 ms, while A10G GPU takes 2.21 ms. We can achieve a lower inference latency by quantization [32] and deploying the 8-bit integer (INT8) inference on U250 FPGA and A10G GPU. Then, the inference latency reduces to 7.3 ms on U250 FPGA and 1.78 ms on A10G GPU.

Based on the requirements of deterministic E2E inference latency and the initial profiling results of the existing solutions, several research questions arise as follows.

1) What are the limitations of the existing acceleration platforms in satisfying the low-latency demands?
2) With quantization optimization, do we have a better computing solution to achieve lower latency than FPGAs and GPUs?[1]
3) If so, how to achieve that?
4) Can we also improve the accuracy with integer quantization?

Our answer to the second question is "Yes." We propose the EQ-ViT architecture and our implemented EQ-ViT design on the AMD Versal adaptive compute acceleration platform (ACAP) VCK190 achieves a latency as low as 0.56 ms, which has 3.2× latency improvement over A10G GPU and 13.1× over U250 FPGA. However, achieving latency as low as 0.56 ms on the heterogeneous Versal ACAP system-on-chip (SoC) involves a lot of design efforts. To ease the programming efforts, we propose the EQ-ViT design automation framework to perform the design space exploration and automatic code generation to facilitate the implementation. In addition, we propose the EQ-ViT algorithm to improve the inference accuracy after the INT8 quantization and EQ-ViT algorithm-hardware co-design to meet the hardware constraints without hurting the algorithm accuracy. Our contributions are summarized below.

1) *Detailed Profiling and Bottleneck Analysis:* To understand the performance constraints of the existing solutions, we perform in-depth kernel-level performance profiling of ViTs on FPGA, GPU, and ACAP in Section II. Based on the bottlenecks for the existing solutions, we propose our solution principles.
2) *EQ-ViT Accelerator and Mapping:* We propose a novel spatial and heterogeneous accelerator template and programming mapping solution to take advantage of the ACAP heterogeneous features: the coexistence of FPGA and artificial intelligence engine (AIE) vector cores on the same SoC in Section IV. Our accelerator architecture features multiple spatial accelerators to improve the AIE core utilization and fine-grained pipeline to overlap the execution time of the accelerators that run on the FPGA and AIEs of the ACAP.
3) *EQ-ViT Algorithm and Algorithm-Hardware Co-Design:* On the algorithm level, we develop a full quantization-aware training (QAT) strategy, the EQ-ViT algorithm, to quantize both the weights and activations into 8-bit integers in Section V. This method improves accuracy on all the four different ViT models. More importantly, our proposed EQ-ViT algorithm-hardware co-design quantizes the nonlinear functions with the algorithm optimization and realizes the efficient hardware implementation for the Softmax and GeLU.
4) *EQ-ViT Automation and System Implementation:* We design EQ-ViT automation framework to implement the EQ-ViT architecture for the four different ViT models on the AMD Versal ACAP VCK190 board. Experiments

[1]Note that, <1 ms latency requirement in the example discussion is for the illustration purposes. The latency requirements differ across various application scenarios. We desire a solution that achieves lower latency than GPUs and FPGAs under the same throughput requirement or achieves higher throughput (or energy efficiency) than GPUs and FPGAs under the same latency requirement. In this article, we discuss such a solution EQ-ViT.

TABLE I
HARDWARE SPECIFICATION COMPARISONS ON PEAK PERFORMANCE FOR DATA TYPES FP32 AND INT8, ON-CHIP MEMORY SIZE, OFF-CHIP BANDWIDTH (BW), TDP AMONG AMD FPGA U250, NVIDIA GPU A10G, NVIDIA GPU JETSON AGX ORIN, AND AMD VERSAL ACAP VCK190

Hardware Spec.	Tech. Node	FP32	INT8	Off-chip BW	On-chip Mem.	Off-chip Mem.	TDP
AMD FPGA U250	16nm	1.2T	6.95T	77GB/s	53MB	16GB	225W
Nvidia GPU A10G	8nm	35T	140T	600GB/s	14MB	24GB	300W
Nvidia GPU Jetson Orin	8nm	5.3T	85T	204GB/s	6MB	64GB	60W
AMD ACAP VCK190	7nm	6.4T	102T	25GB/s	33MB	8GB	<180W

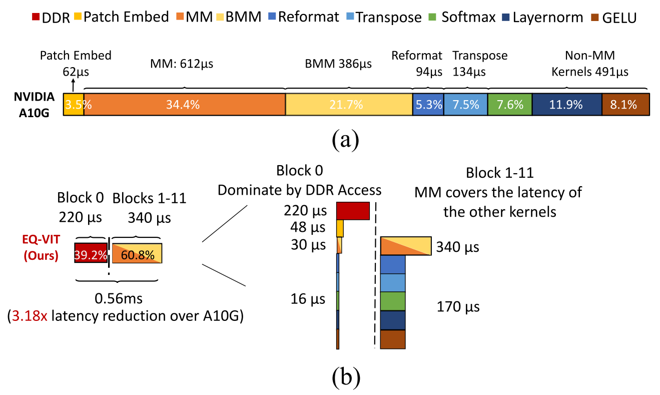

Fig. 2. E2E inference latency comparison of using TensorRT on NVIDIA A10G GPU and using EQ-ViT (ours) on AMD Versal VCK190 ACAP for the representative ViT model DeiT-T with INT8 precision when batch size = 6. (a) DeiT-T INT8 E2E latency on A10G is 1.78ms, (b) DeiT-T INT8 E2E latency on EQ-ViT (Ours) is 0.56ms.

in Section VI show EQ-ViT achieves accuracy improvement with 2.4% and average speedups of up to 315.0, 3.39, 3.38, 14.93, 59.5, 13.1× over computing solutions of Intel Xeon 8375C vCPU, A10G, A100, Jetson AGX Orin GPUs, AMD ZCU102, and U250 FPGAs.

5) *EQ-ViT Generality Discussion:* We discuss how EQ-ViT mapping framework can be applied to the other architecture, e.g., FPGA and GPU, to improve the performance in Section VII. We further discuss the microarchitecture insights, i.e., what role reconfigurability plays in the future heterogeneous architecture.

II. BOTTLENECK ANALYSIS AND PROPOSED SOLUTION

In this section, we first explain the performance bottlenecks of the current solutions on FPGA, GPU, and ACAP. Then, we discuss our proposed design principles.

First, FPGAs are mainly constrained by the limited computation resources. Table I indicates that AMD FPGA U250 (Ultrascale+, 16 nm fabrication) has the lowest peak performance among the three hardware platforms, at 1.2 TFLOPS for FP32 and 6.95 tops for INT8 under 250 MHz. When transitioning from FP32 to INT8, the E2E latency decreases from 50.3 to 7.3 ms. However, both the cases are computation-bound and latency can not be further reduced because of the limited computation resources from DSP/LUT in FPGA.

GPUs have abundant computation cores, e.g., NVIDIA introduces Tensor cores since the volta architecture. Table I reveals that GPU A10G (the ampere architecture, 8 nm fabrication) boasts the highest peak performance at 35 TFLOPS for FP32 and 140 TOPS for INT8. Tools like TensorRT simplify inference streamline through the methods, such as quantization. However, Fig. 1 shows that the E2E latency on GPU A10G is 2.21 ms for FP32 and 1.78 ms for INT8. This results in a modest 1.24× E2E improvement, significantly smaller than the theoretical peak computation performance improvement from FP32 to INT8 (4×, calculated as 140T/35T). To understand the performance bottleneck, we utilize NVIDIA Nsight System [33] and depict the kernel-level time breakdown for INT8 in Fig. 2. We identify the following performance constraints for using TensorRT on the GPU: ❶ *Low Tensor Cores Utilization for INT8 MM Kernels:* Although MM kernels constitute 34.4% of the total runtime, their effective throughput is 23 tops, representing only 16% utilization of the peak INT8 computation performance for GPU A10G. ❷ *TensorRT Adopts an Implicit Quantization Policy, Which Leads to BMM Computing in FP32, Not in INT8:* Quantization enables MM and batch-MM (BMM) to compute in INT8 for higher throughput. However, according to the NVIDIA Nsight compute kernel-level profiling report, BMM kernels compute in FP32. This is related to the implicit quantization strategy applied by TensorRT [34], which will quantize one kernel only when this kernel runs faster in INT8. Otherwise, TensorRT will assign a higher precision to this kernel, FP32, by default. Despite having only 1/6 of the total operations of MM kernels, BMM kernels contribute to 21.7% of the total runtime. We calculate their effective throughput as 6.3 TFLOPS, which is 18% of the peak FP32 computation performance for A10G. ❸ *The Data Type Conversion Between FP32 and INT8 Consumes Non-Negligible GPU Cycles:* MM kernels are processed in INT8 mode using NVIDIA Tensor cores, while other kernels use FP32 mode with NVIDIA CUDA cores. Data type conversions between FP32 and INT8, known as Reformat are introduced. This operation is significant, accounting for 5.3% of the E2E latency. ❹ *The Nonlinear Kernels Take Significant GPU Cycles:* Non-MM kernels, such as Softmax, GeLU, and LayerNorm, collectively contribute 27.6% of the total, despite their operations being only 1.5% of MM kernels. This is due to these kernels involving special functions, such as exponent functions, division, and square root.

AMD Versal ACAP is a heterogeneous SoC, featuring ARM CPUs, FPGA, and AIE vector cores. The AIEs support several data types, including FP32, INT16, and INT8 [35]. ACAP integrates the aspects of both the domains, that is, FPGA for reconfigurability and AIEs for abundant computation cores. We deployed the DeiT-T model FP32 version on the VCK190 board using CHARM [36], an SOTA deep learning inference framework on the ACAP architecture. Fig. 1 illustrates that CHARM has an E2E latency of 48.07 ms, which is 27× slower than using TensorRT on GPU A10G under FP32. This performance lag is mainly due to the significant load/store of the feature data from/to off-chip memory, caused by the FP32 model's size exceeding the VCK190 on-chip storage capacity of 33 MB. Quantizing the model into INT8 allows

TABLE II
ARCHITECTURE AND ALGORITHM FEATURES OF EQ-VIT AND COMPARISONS WITH PRIOR WORKS

Prior Works	Computing Platform		Accelerator Features						Algorithm & Algorithm-Hardware Co-Design Feat.		
	Board Type	GOPS/(GB/s)	Multi Spatial Accelerators	Hardware Specialization	On-chip Forwarding	Fine-grained Pipeline	Explicit Quant.	Compute Util.	Activation-aware Quant.	Nonlinear Quant.	Accuracy Gain
TensorRT [31]	GPU	–	×	×	×	×	×	Low	–	–	–
Herald [37]	ASIC	–	✓	×	×	×	–	High	–	–	–
MAGMA [38]	ASIC	–	✓	✓	×	×	–	High	–	–	–
ViA [39]	FPGA U50	372/316=1.18	✓	✓	×	✓	×	High	–	–	–
CHARM [36]	ACAP VCK190	6400/25.6=250	✓	✓	×	✓	×	High	–	–	–
ViTCoD [40]	ASIC	256/76.8=3.3	×	✓	×	×	×	High	×	×	×
HeatViT [30]	FPGA ZCU102	1260/19.2=65.6	×	✓	×	×	✓	High	×	✓	×
Auto-ViT-Acc [41]	FPGA ZCU102	1260/19.2=65.6	×	✓	×	×	✓	High	×	×	×
SSR [42]	ACAP VCK190	102,400/25.6=4000	✓	✓	✓	✓	✓	High	×	×	×
EQ-ViT (Ours)	ACAP VCK190	102,400/25.6=4000	✓	✓	✓	✓	✓	High	✓	✓	✓

Note: [31], [36]–[39] are architecture and mapping frameworks. [30], [40], [41] and EQ-ViT (ours) are algorithm-hardware co-design frameworks.

it to fit on-chip. However, without careful design, ACAP acceleration may face similar limitations (from ❶ to ❹) as A10G, and potentially worse due to VCK190's limited 4.2% off-chip BW compared to A10G. This leads to the following question. How can we optimize latency for INT8 ViT on ACAP, given its high computation intensity but constrained off-chip BW?

Proposed Design Principles: We propose EQ-ViT to optimize latency for INT8 ViT, which circumvents all the constraints from ❶ to ❹ typically encountered in GPU. The key idea of EQ-ViT is to design multiple heterogeneous MM accelerators on AIEs, design other non-MM kernels on FPGA, and overlap the execution of kernels running on AIEs and FPGA. Fig. 2(b) demonstrates the kernel runtime overlapping in EQ-ViT. However, new challenges appear. *First*, we need to enable explicit INT8 computation for BMMs and achieve high computation utilization for both MMs and BMMs. The computation and communication requirements of MMs and BMMs are different. Overlapping these two types of kernels can improve both the computation and communication utilization. *Second*, we need to design efficient accelerators for nonlinear kernels (Softmax, GeLU, and LayerNorm). *Third*, we need to leverage the flexible on-chip memory architecture provided by FPGA on ACAP to enable the data forwarding in the adjacent kernels and further reduce the off-chip memory access. *Fourth*, we need to carefully overlap the execution time and optimize workload partitioning and resource partitioning jointly, for utilization optimization, high throughput, and low latency. *Fifth*, we need analytical models to optimize the E2E latency under computation resource and communication bandwidth constraints. *Sixth*, we need to keep the accuracy after quantization and, if possible, enhance it.

III. BACKGROUND AND RELATED WORK

In this section, we first discuss the background for the ViT model architecture, and the existing quantization methods for ViT in Section III-A. In Section III-B, we discuss prior works on the hardware acceleration and mapping frameworks on ASICs, FPGAs, GPUs, and ACAP. We also discuss the algorithm-hardware co-design frameworks. We summarize our proposed methodologies in hardware accelerator architecture and the algorithm with the prior works in Table II.

A. Vision Transformer

Transformers were initially proposed to handle the learning of long sequences in NLP tasks. Great interest has surged following the work [1] that applies a transformer architecture for the image classification without reliance on the convolutional architectures (CNN). With more data, the data enhancement techniques or extended training epochs, ViTs can achieve significantly improved task accuracy [2]. Currently, ViTs excel over CNNs in terms of both the speed and accuracy in various computer vision tasks, including image classification [15], object detection [43], and real-time object detection [44].

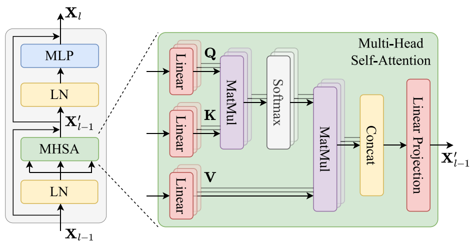

Fig. 3. Computation flow of one transformer encoder.

ViT Architectures: The input image is first divided and arranged into a sequence of patches (or tokens). This sequence is then passed through an L-layer Transformer encoder [45]. Each Transformer layer/block consists of two main components (Fig. 3): 1) a multihead self-attention (MSA) module and 2) a multilayer perceptron (FFN) module. For instance, the DeiT-T model is composed of $L = 12$ Transformer blocks, where the typical input image resolution is 224×224 with a patch size of 16×16. Consequently, this results in a sequence of $n = 196$ tokens, each token being embedded with 64×3 dimensions and utilizing $h = 3$ heads, and dim $= 64$ per head.

Quantization on Transformers: Quantization is one of the most powerful ways to decrease neural networks' computational operations and memory footprint. Current quantization methods can be divided into two categories: 1) QAT [46] and 2) posttraining quantization (PTQ) [47]. NLP-oriented Transformers mainly employ PTQ for the two reasons [48], [49], [50]: QAT needs the open dataset. If the dataset is not publicly available, users have to use PTQ. QAT requires significant computational resources to support the training of large model sizes (usually over 350M), to which academics usually have limited access. However, the compact model size of ViT and the presence of the public datasets make it a suitable candidate for QAT, thereby sidestepping the notable accuracy decrease that is often associated with PTQ. [51] proposes a QAT method for ViTs with information rectified. However, this work does not quantize the nonlinear operations, which causes more hardware overhead because of the data conversion between different data types (dequantizing and requantizing), and etc. Moreover, several existing works [30], [52], [53], [54] utilize *model pruning or sparsity* to reduce the computation operations in ViTs. However, these techniques often lead to unavoidable accuracy drops. In EQ-ViT, we aim to implement a fully quantized ViT through the QAT algorithm and to improve the accuracy.

B. Transformer Accelerators on Hardware

Hardware acceleration for neural networks spans various platforms like ASICs, GPUs, FPGAs, and ACAPs as shown in Table II. ACAP stands out with its high theoretical INT8 performance but faces a challenge with its relatively low off-chip bandwidth. This requires more design efforts due to the high computation-to-communication (CTC) ratio on ACAP. Nevertheless, EQ-ViT incorporates all the listed accelerator and algorithm-hardware co-design features, achieving the highest computation utilization and the lowest latency for ViT compared to the existing works.

Hardware Acceleration and Mapping Framework: TensorRT [31] provides a general quantization solution on GPUs. However, TensorRT adopts an implicit quantization policy and faces low INT8 tensor core utilization due to its sequential execution model, i.e., calling each kernel one after another. Herald [37] introduces a heterogeneous system with simultaneous spatial accelerators (accs), allowing for optimization exploration as different accs may have varied CTC ratios. While Herald integrates well-designed accs, EQ-ViT goes a step further by supporting the accs hardware specialization and jointly optimizing accs scheduling and designing. MAGMA [38] proposes an automatic framework for the multitenancy heterogeneous architectures but suffers from significant latency due to the off-chip communication. This is not ideal for scenarios that are sensitive to time. In contrast, EQ-ViT customizes on-chip forwarding among any two adjacent accs to optimize the off-chip access. ViA [39] applies a well-customized spatial solution on U50 FPGA, supporting at most two spatial accs, while EQ-ViT explores more accs. FLAT [55] applies a tensor fusion mechanic and a tiling method to reduce the communication in attention-based models. CHARM proposes an open-source framework that composes multiple specialized accelerators, but it only supports FP32 data type and falls short of meeting real-time requirements on ACAP. EQ-ViT features a spatial architecture with customized accs. The fine-grained pipeline structure and on-chip data forwarding achieve deterministic low latency.

Algorithm-Hardware Co-Design Acceleration for ViT: ViT architecture works [30], [40], [41] also consider algorithm adaption, e.g., sparsity, to speed up the model inference. ViTCoD [40] efficiently prunes and polarizes attention maps to create denser or sparser fixed patterns, reducing attention computations. HeatViT [30] employs image-adaptive token pruning and 8-bit quantization to eliminate the model redundancy, resulting in improved on-device throughput. Auto-ViT-Acc [41] utilizes network search to tune the quantization choices for the best latency under the frame-per-second (FPS) performance constraints. SSR [42] provides a framework that explores the latency throughput tradeoff for the transformer-based applications. While enabling the hardware accelerator features, there is a lack of discussion about the algorithm design and the algorithm-hardware co-design features. However, these works have two main limitations.

1) In [40] and [41], the nonlinear operators in ViT models are computed in FP32, leading to significant hardware overhead. HeatViT [30] uses polynomial approximations for GeLU and Softmax, quantizing them into INT8. However, this approach consumes a significant amount of FF/LUT resources due to the exponent “e”' in Softmax. EQ-ViT (ours) employs “2” as the exponent, resulting in lower FF/LUT resource usage.
2) Task accuracy degrades. ViTCoD applies uniform pruning pattern to compress the attention matrix, leading to accuracy drops of 0.5%∼1%. HeatViT and Auto-ViT-Acc fail to consider the inherent data distribution within ViTs, resulting in inconsistencies between the quantization strategy and the data distribution. In contrast, EQ-ViT introduces a hardware-efficient nonlinear quantization and achieves better task accuracy than the full-precision models through the activation-aware quantization.

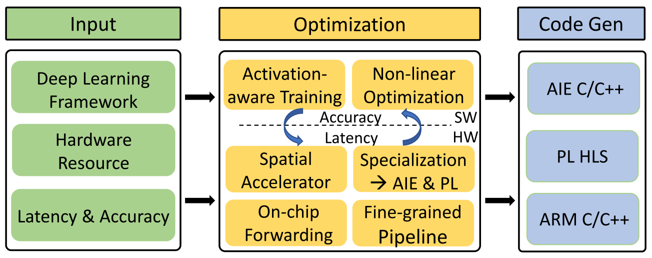

Fig. 4. EQ-ViT software/hardware co-design framework.

IV. EQ-ViT Framework and Architecture

In this section, we first illustrate the proposed framework and the EQ-ViT heterogeneous accelerator. We then elaborate on the detailed mapping methodology.

A. EQ-ViT Framework Overview

Our EQ-ViT provides the optimization for the algorithm/hardware co-design. In Fig. 4, our framework takes the latency and accuracy requirement and the hardware information from the user. These combined constraints will decide the final quantization strategy by the activation-aware training and mapping strategy through (1)–(7) in Section IV-D. Given an application, our EQ-ViT will conduct activation-aware training and provide accuracy under 32, 16, 8, and 4 bits for both the activations and weights. Then, according to the accuracy constraint and the hardware information, EQ-ViT will pick a quantization strategy that meets the accuracy requirement while best fitting the vector processors (AIEs). For instance, Versal VEK280 provides peak performance under the 8 bits×4 bits mode whereas VCK190 provides peak performance under the 8 bits×8 bits mode. Then, we use (1)–(7) to optimize the throughput under the latency constraint and the quantization strategy. If the model quantization is insufficient to target a single board, our work can be used in concert with partitioning approaches to map larger models onto the multiple devices [23]. Our EQ-ViT framework also includes a Python-based code generation toolflow. Based on the generated mapping strategy, it can instantiate the code template to generate the design source files, including ARM CPU host code, FPGA high-level synthesis code, and AIE intrinsic C/C++ code which can be directly compiled and deployed on Versal ACAP.

B. EQ-ViT Heterogeneous Accelerator Overview

Fig. 5 shows the overall EQ-ViT architecture on ACAP. It is composed of multiple spatial accelerators with MM units

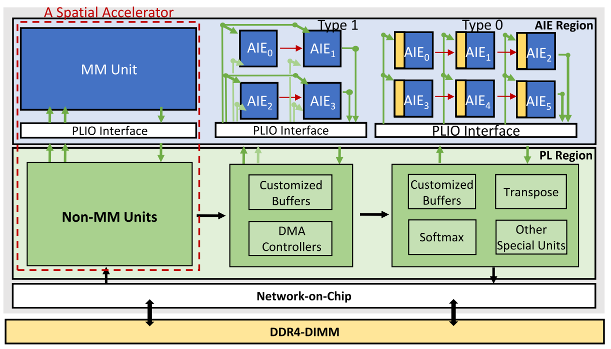

Fig. 5. Proposed EQ-ViT architecture overview.

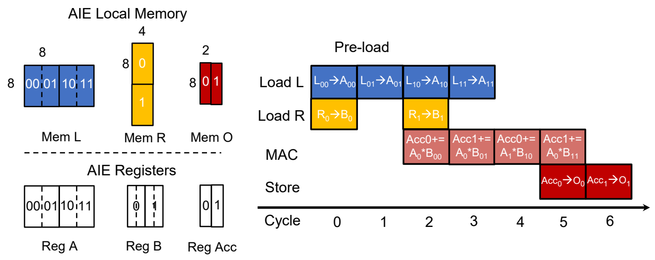

Fig. 6. Efficient single AIE design.

allocated to the AIE region and non-MM units allocated to the PL region. The MM and non-MM units are connected through the PLIO interface. We design specialized MM units for the computation-intensive kernels, e.g., MM, BMM, and Conv by exploring 3-D parallelism on the AIE array. By leveraging the flexibility of the PL region, we implement non-MM units for transpose, Softmax, Layernorm, and GeLu. Based on these building blocks, our proposed EQ-ViT architecture has the following hardware characteristics: 1) we apply *spatial architecture* that multiple accelerators compute different kernels with high AIE utilization at the same time instead of using one unified accelerator and launching it sequentially; 2) to reduce the expensive off-chip memory access, we explore the *on-chip data forwarding* between different spatial accelerators; and 3) we propose a *fine-grained pipeline* structure within each spatial accelerator to further overlap the execution of nonlinear and element-wise kernels with MMs to reduce latency. The details will be elaborated in Section IV-C.

C. Hardware Design Methodology

High Utilization Matrix Multiply Design on Single AIE and AIE Array: When designing the MM/BMM kernels under the INT8 data type, efficient communication between the PL SRAM, AIE local memory, and registers is important to saturate the abundant computation resource. We optimize MM/BMM kernels from the two levels, the single AIE and AIE array levels.

In the single AIE level, based on the byte-level flexibility of AIE, we write efficient AIE intrinsic instructions to make full use of the 2 Kb vector register to sustain the 128 MACs/cycle throughput with two 256 bits/cycle load instructions. The 128 MACs can be constructed as a 16×8 MAC array where the second dimension is the reduction dimension. Under the constraints of 2 Kb vector register as well as the 256 bits/cycle load bandwidth, we customize the 128 MACs into an 8×8×2

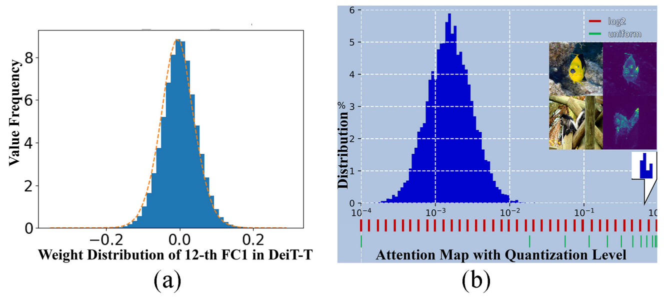

Fig. 7. Data distribution in DeiT-T. (a) A representative normal distribution of the weight of the 12th FC1 layer. (b) Long-tail distribution for attention map.

3D-SIMD instruction. Based on our atomic 8×8×2 MAC operation, the execution pipeline of a MM with size 8×16×4 is shown in Fig. 6. In order to achieve back-to-back issued MAC instructions, we allocate 8×8 and 8×4 8 bits vector registers and use the double buffer technique to hide the latency of loading from the local memory to the vector registers. After two cycles of preloading the data into AIE registers for the LHS and right-hand-side (RHS) operands, the MAC operations can be issued without the idle cycles. Based on this scheduling, it can also handle the MM with a larger size at the expense of only two preload cycles.

When scaling out to the AIE array, the shape variance of the multiple layers within a transformer block often leads to the hardware underutilization [36], [37], [56]. Thus, for each layer within a transformer block, we design a customized MM unit that perfectly matches the shape of the layer. The number of AIEs utilized in each MM unit are proportional to the total number of operations within the layer. We propose two kinds of MM units as shown in Fig. 5. For AIEs of Type 0 that take both the activation and weights as their operands, we efficiently allocate the AIE local memory to make sure the weight of all the blocks fit and loaded during initialization without further excessive loads. Thus, it saves the PLIO of sending the RHS operands (weights). For the kernels that the weights cannot fit in the AIE local memory or the two operands are both activations (attention batch dot), we map them to AIE design of Type 1.

Element-Wise and Nonlinear Kernel Design: Element-wise kernels and nonlinear kernels, including Transpose, VectorAdd, Reformat, Softmax, LayerNorm, and GeLU account for less than 2% of the total operations. However, they collectively contribute 40% of the total execution time as shown in Fig. 2. To overlap the latency of these operations with the MM operations, we apply a similar line-buffer methodology proposed in SSR [42] to enable a fine-grained pipeline. Beyond the proposed method, we further apply quantization to the nonlinear kernels introduced in Section V-C, significantly reducing the number of resources used in the PL.

D. Hardware Design Optimization

We mathematically formulate a mixed-integer-programming (MIP) [57] optimization problem to guide the design space exploration and determine the hardware resource partitioning and configuration for each spatial accelerator. We denote the number of accelerators and batches as Acc and B. The ViT graph is denoted as G and the start execution time of each node included in the graph is referred to as T_n. $D_{n,m}$ refers to a binary dependency matrix of the nodes in the graph, where $D_{n,m} = 1$ means node, m depends on n. $E_{n,a}$ and $A_{n,a}$ are the integer and binary matrix variables representing the execution time and allocation map of each node on every accelerator. (2) limits the finish time of every node in batch 1 as the latency of the first batch should meet a certain budget, e.g., *Budget* as 1 ms. The goal is to maximize the overall throughput calculated as (1) and (3); (4) and (5) guarantee each node will be mapped to only one accelerator and each time one hardware accelerator will only execute one logic node in the graph. The execution order should follow the dependency map (6). The sum of hardware utilization should meet the hardware constraints (7)

$$\text{maximize} \quad B/\text{Lat}_{\text{all}} \tag{1}$$

$$\text{s.t.} \quad T_n + E_{n,a} \times A_{n,a} \leq \text{Budget} \ \forall n \in (G_1) \tag{2}$$

$$\text{Lat}_{\text{all}} = T_n + E_{n,a} \times A_{n,a} \ \forall n \in (G) \tag{3}$$

$$\Sigma_{a=1}^{\text{Acc}} A_{n,a} = 1 \ \forall n \in G \tag{4}$$

$$\begin{aligned} &T_m \geq T_n + E_{n,a} \times A_{n,a} \ or \ T_n \geq T_m + E_{m,a} \times A_{m,a} \\ &\quad \forall (n, m) \in G, \forall a \in \text{Acc}, D_{n,m} = 0, A_{m,a} = A_{n,a} \end{aligned} \tag{5}$$

$$T_m \geq T_n + E_{n,a} \times A_{n,a} \ \forall (n, m) \in G, D_{n,m} = 1 \tag{6}$$

$$\begin{aligned} &\Sigma U_{\{\text{RAM,AIE,PLIO,DSP}\}a} \leq HW_{\{\text{RAM,AIE,PLIO,DSP}\}} \\ &\quad \forall a \in \text{Acc}. \end{aligned} \tag{7}$$

V. EQ-ViT Algorithm

In this section, we first probe into a comprehensive analysis of the data distribution (weight and activation) of ViTs and arrive at several discoveries. Then, we develop activation-aware QAT to quantize ViTs and improve accuracy. Furthermore, we propose INT-Softmax$_{2^n}$ and I-GeLU$_{\text{Imp}}$ to reduce the hardware resources.

A. Discovery of Data Distribution Within ViTs

Weight: Data follows a standard normal distribution [Fig. 7(a)].

Activation: Two key features impact the quantization strategy, *long-tail distribution* and *channel-wise outliers*.

Long-Tail Distribution:

Attention Map: The attention map is the feature map of the Softmax output. To preserve the informative message of the Softmax, we plot attention maps in the real and log domain [Fig. 7(b)], which reveals a long-tail distribution. Compared to the uniform quantization (with 8-bit), which assigns only one bin to such a large number of values, the log2 method has more resolution (24 bins) to cover this data range. This indicates that the low-bit log2 method plays an ideal quantization choice.

Channel-Wise Outliers:

Large Interchannel Variations in the Residual Link Addition: As shown in Fig. 8(b), the channel-wise ranges in ViTs exhibit more significant fluctuations than in ResNets. As the channels with outliers require larger scales than the others, using common quantization methods like the layer-wise quantization with the same parameters for all the channels would result in an unacceptable quantization error.

Systematic and Fixed Outliers: Although outliers appear in every sequence, they are concentrated in fixed channel dimensions of the residual link addition as shown in Fig. 8(a)

B. Activation-Aware QAT

We propose two novel quantization methods, *long-tail-oriented quantization* and *outlier-predictable QAT*. Assuming the bit-width is b, the quantizer $\text{Q}(X|b)$ can be formulated by mapping a floating-point number $X \in R$ to the nearest quantization bin. Among various quantizers, uniform [59] and log2 [60] are typically used. Apart from the special data distribution in Section V-A, we apply the layer-wise uniform quantization on the weights and activations.

1) Long-Tail-Oriented Quantization: Log2Q on Attention Map: Based on Section V-A, we apply Log2Q on the attention map to preserve the informative content as

$$\text{Attn}_Q = \text{Log2Q}(\text{Attn}|b) = \text{clip}\Big(\lfloor -\log_2(\text{Attn}) \rceil, 0, 2^b - 1\Big). \tag{8}$$

2) Outlier-Predictable QAT: We propose the *outlier-predictable training* that obtains the precise channel indices of outliers in the addition of residual links and regularizes scales of outliers with different *power-of-two coefficients* (PTCs) in channel wise.

PTCs on the Residual Link Quantization: Given the input activation (token) $X \in B \times L \times C$ (B: batch size, L: token/sequence length, C: the channel dimension of one token, and the PTC $r \in \mathbb{N}^C$, then the quantized activation X_Q is

$$X_Q = \text{Q}(X|b) = \text{clip}\left(\lfloor \frac{X}{2^r s} \rceil + z, 0, 2^b - 1\right) \tag{9}$$

$$s = \frac{\max(X) - \min(X)}{2^R\big(2^b - 1\big)}, \quad z = \text{clip}\left(\left\lfloor -\frac{\min(X)}{\max(X)} \right\rceil, 0, 2^b - 1\right) \tag{10}$$

where the outlier channel index is i, PTC is $r \in [2,3,4]$, s is the scaling factor, and z is the zero-point.

Outlier-Predictable Training: It includes three stages: 1) initialize the PTC with the full-precision model estimated by threeσ method [62]; 2) search for the channel index i and the PTC r with the l_2 regularization; and 3) fix the index i and r obtained in stage 2 and fine tune the model.

C. Nonlinear Operations Quantization

1) INT-Softmax$_{2^n}$: We replace the natural constant e inside the Softmax with the power of 2 [63] with the integer inputs. i represents the ith token

$$\text{INT-Softmax}_{2^n}(X) = \frac{\exp(X_i)}{\Sigma_{l=1}^{L}\exp(X_l)} \rightarrow \frac{2^{X_i}}{\Sigma_{l=1}^{L} 2^{X_l}}. \tag{11}$$

Log2Q With INT-Softmax$_{2^n}$: Similar to [64], we utilize Log2Q on the attention map. We then integrate the power of 2 inside the Softmax and the operation can be modified as

$$\begin{aligned} \text{Attn}_Q &= \text{Log2Q}(\text{Attn}|b) \\ &= \text{clip}\Big\lfloor -\log_2 \Sigma_{l=1}^{L} 2^{\hat{X}_l} + \hat{X}_i \Big\rceil, 0, 2^b - 1. \end{aligned} \tag{12}$$

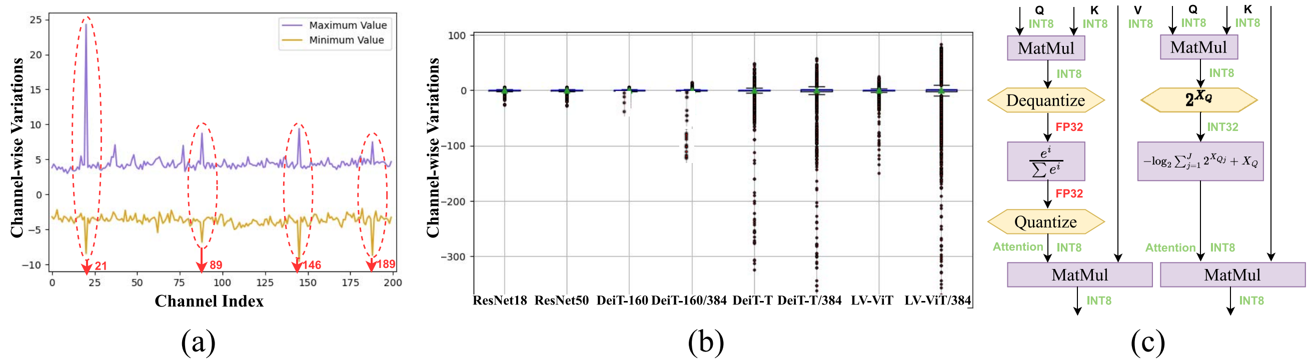

Fig. 8. (a) Channel-wise minimum and maximum values of the second residual link addition in the 9th block of DeiT-T. (b) Channel-wise ranges of the last residual link addition in representative models. (c) Comparison of common INT-Softmax [61] and INT-Softmax$_{2^n}$ in quantized MSA inference.

TABLE III
MODEL STRUCTURES OF FOUR DIFFERENT VIT MODELS

Model	#Head	Embed. Dim	Depth	Precision	Model (MB)	MACs (G)
DeiT-T	3	192	12	INT8	5.6	1.3
DeiT-160	4	160	12	INT8	4	0.9
DeiT-256	4	256	12	INT8	7.4	2.1
LV-ViT-T	4	240	12	INT8	6.75	1.6

The exponent function is a crucial component of the Softmax, but its nonlinearity makes it expensive to implement on the hardware. Combined with the Log2 quantization, the Softmax function can be executed with only addition computation and removes division thus can be implemented by LUTs on FPGA instead of AIEs. As shown in Fig. 8(c), the floating-point exponential calculation of the INT-Softmax$_{2^n}$ is replaced with BitShift and addition and keeps integer-only data type.

2) I-GeLU$_{Imp}$: We adapt I-GeLU [65] to a combination with linear kernels and lookup table under INT8 mode, since $1+L(x)$ is an odd function within the range (0, 2)

$$\text{I-GeLU}_{\text{Imp}} = \begin{cases} 0 & \text{if } -\left(2^8-1\right) \le x \le -3 \\ \{0,0,0,0,1\} & \text{if } x \in \{-2,1,0,1,2\} \\ x & \text{if } 3 \le x \le 2^8-1. \end{cases} \tag{13}$$

For implementation, we preload the requantized integer value directly on-board as (13).

VI. EXPERIMENT

A. Experiment Settings

Application and Training Framework Setup: Our experiments are conducted on the ImageNet-1k [66], Cifar-100, and Cifar-10 [67] datasets in PyTorch 3.8. We use two representative ViTs, DeiT [2], and LV-ViT [68], in Table III. The baseline models with FP32 are obtained from the TorchVision. The outlier-predictable training follows Q-ViT with distribution-guided distillation (DGD) techniques [51], and the training process is executed on four NVIDIA V100 GPUs. We set stage 1 to 70 epochs and stage 2 to 30 epochs.

Hardware Setup: We evaluate EQ-ViT the on AMD ACAP VCK190. We compare EQ-ViT with the other SOTA implementations on CPU, FPGA, and GPU. For each model, we iterate the inference for over 60 s and perform this measurement ten times to calculate the average inference latency. On CPU, we measure the inference latency on an m6i.large instance from Amazon AWS using Pytorch 2.0.1. The instance has two Intel Xeon 8375C vCPU cores running at 2.9 GHz and thermal design power (TDP) is 300 W. On GPUs, we measure the performance of TensorRT [31] on A10G (8 nm), A100(7 nm), and Jetson AGX Orin (8 nm). We first use onnx 1.14.0 to compile the PyTorch model into the onnx format, then use TensorRT 8.6 and its Python interface to compile the onnx model into the TensorRT engine. To perform the INT8 inference, we enable the *tensorrt.BuilderFlag.INT8* flag in compilation. The power consumption of the GPUs is measured via NVIDIA-smi [69]. For the CPU and GPU experiments, the PyTorch models are from the meta research [70].

On FPGA, we compare EQ-ViT with HeatViT [30] on AMD Zynq ZCU102 and AMD Alveo U250. We compare EQ-ViT with SSR [42] on the same device VCK190. We measure the power of VCK190 using the AMD board evaluation and management [71]. To be noted, EQ-ViT provides the algorithm and the algorithm/hardware co-design to explore different quantization strategies, e.g., activations 8 bits and weights 4 bits (A8W4) without the accuracy loss. We add the new estimated results (est.) in Table IV when using the A8W4 quantization on AMD Versal VEK280 which provides 4× 8 bits × 4 bits MAC operations/cycle/AIE over VCK190 with 8 bits × 8 bits precision. Our estimation shows that EQ-ViT further reduces the latency by 1.67× using VEK280 over VCK190. This gain can not be achieved without the algorithm and the algorithm/hardware co-design, demonstrating the key new contribution of EQ-ViT.

B. ViT Inference Performance and Energy Efficiency Analysis

① Performance and Energy Efficiency Comparison Among CPU, GPU, FPGA, and ACAP: We apply our EQ-ViT framework to four different ViT applications under the INT8 quantization mode and evaluate the on-board implementation on AMD Versal VCK190. We compare EQ-ViT with six works on CPU, GPUs, and FPGAs regarding latency and energy efficiency on the four models in Table IV. Here, we report the performance when setting the latency budget as 1 ms. EQ-ViT DSE finds the optimal throughput design under this latency constraint when the batch size is set to 6. The achieved latencies are 0.56, 0.46, 0.89, and 0.61 ms for the four applications. In contrast, the solutions on other platforms have larger

TABLE IV
COMPARISON OF EQ-VIT AND WORKS ON CPU, GPU, FPGA, AND ACAP IN LATENCY AND ENERGY EFFICIENCY ON FOUR MODELS

Model	**# of Batch**	**Metric**	**PyTorch Xeon8375 10nm**	**TensorRT A10G 8nm**	**TensorRT A100 7nm**	**TensorRT Orin 8nm**	**HeatViT ZCU102 16nm**	**HeatViT U250 16nm**	**SSR VCK190 7nm**	**EQ-ViT (ours) VCK190 7nm**	**EQ-ViT (ours) VEK280 (est.) 7nm**
DeiT-T	6	Latency (ms)	167.68	1.78	1.84	7.97	32.72	7.3	0.54	**0.56**	**0.33**
		FPS (image/sec.)	36	3371	3260	753	183	822	11111	**10695**	**18010**
		Energy.Eff (FPS/W)	3.8	15.8	18.6	17.7	19.4	10.2	213.7	**224.7**	**427.8**
DeiT-T-160	6	Latency (ms)	129.01	1.78	1.73	7.92	29.75	6.34	0.50	**0.46**	**0.28**
		FPS (image/sec.)	47	3371	3468	758	202	946	11976	**13187**	**21702**
		Energy.Eff (FPS/W)	4.9	16.9	20.0	19.0	21.9	12.2	206.8	**280**	**503.5**
DeiT-T-256	6	Latency (ms)	294.61	2.07	2.09	10.44	39.33	9.13	0.98	**0.89**	**0.53**
		FPS (image/sec.)	20	2899	2871	575	153	657	6122	**6726**	**11393**
		Energy.Eff (FPS/W)	2.2	12.5	15.0	13.2	14.7	8.5	102.9	**142.8**	**269.3**
LV-ViT-T	6	Latency (ms)	213	2.55	2.54	10.1	43.21	9.36	0.85	**0.61**	**0.37**
		FPS (image/sec.)	28	2353	2362	594	139	639	7059	**9836**	**16017**
		Energy.Eff (FPS/W)	3	10.6	12.9	13.5	13.5	7.8	115.3	**202.8**	**359.9**

TABLE V
LATENCY COMPARISON BETWEEN ON-BOARD MEASUREMENTS AND MIP MODELING ESTIMATIONS FOR FOUR VIT MODELS

Model	**# of AIE**	**Estimation**	**On-board**	**Error Rate**
DeiT-T	394	0.58 (ms)	0.56 (ms)	4%
DeiT-160	396	0.48 (ms)	0.46 (ms)	5%
DeiT-256	399	0.92 (ms)	0.89 (ms)	3%
LV-ViT-T	398	0.59 (ms)	0.61 (ms)	-3%

TABLE VI
RESOURCE UTILIZATION OF SOFTMAX AND GELU BEFORE VERSUS AFTER EQ-VIT ALGORITHM CHANGES FOR HARDWARE EFFICIENT IMPLEMENTATION ON VCK190

Operations	**Softmax [36]**	**INT-Softmax(ours)**	**GeLU [36]**	**INT-GeLU(ours)**
REG	62415 (4.17x)	14962	22238 (137x)	162
LUTLogic	94739 (14.48x)	6545	14222 (142x)	100
LUTMem	37668 (18834x)	2	1392 (-)	0
RAM	147 (9.19x)	16	1 (-)	0
DSP	196 (7.00x)	28	128 (-)	0

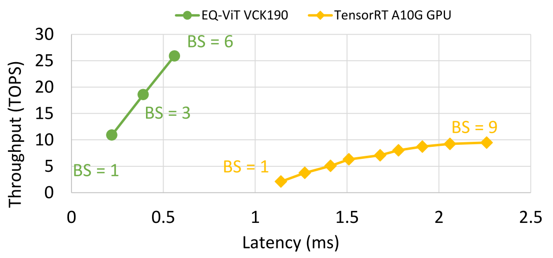

Fig. 9. Latency and throughput tradeoff comparison between EQ-ViT on VCK190 and TensorRT on A10G GPU.

latency and do not meet the latency constraint under the same batch size. For all the four applications, the average latency gains are 315.0, 3.39, 3.38, 14.93, 59.5, and 13.1×, and the gains of energy efficiency are 62.2, 15.33, 12.82, 13.31, 13.5, and 21.9× when comparing to the Intel Xeon 8375C vCPU, A10G, A100 GPUs, AMD ZCU102, and U250 FPGAs. We further analyze the latency improvement from the four features (4.2x, 3.4x, 2.3x, and 2.7x) in Section VIII, together achieving 89× latency reduction from 50 ms using the FP32 model with CHARM to 0.56 ms using the INT8 model with EQ-ViT on VCK190. We also applies the int8 GEMM solution proposed by [35]. For DeiT-T with batch equals 6, it achieves 12.1 ms latency as it only implement a monolithic accelerator and requires the weights and activation to be accessed from the off-chip memory. By applying the on-chip data forwarding, fine-grained pipeline and multiple spatial accelerators, EQ-ViT achieves 21.6× performance improvement.

②*Analytical Model Versus EQ-ViT On-Board Implementation:* We evaluate the latency of the four ViT models on AMD Versal VCK190 and compare them with the proposed MIP modeling. Guided by the MIP, all the four cases utilize over 98.5% AIE. The error rate in percentage refers to the difference between the estimated latency by MIP and our real on-board implementation. On average, the MIP modeling achieves a high prediction accuracy and has less than 4% error rate as shown in Table V.

③*The Effect of Batch Size on Latency-Throughput Tradeoff:* We can leverage the MIP-based analytical model to perform the latency-throughput tradeoff in EQ-ViT, e.g., find the designs that achieve the highest throughput under the latency constraints. Fig. 9 shows the latency-throughput Pareto fronts of EQ-ViT on VCK190 and TensorRT on A10G GPU. EQ-ViT achieves a better Pareto front than that of GPU.

④ *Resource Utilization Before Versus After EQ-ViT Hardware-Efficient Algorithm Adaption for Two Non-MM Kernels Softmax and GeLU:* We compare the hardware utilization of the optimized Softmax and GeLU implementation with the previous FP32 design reported in CHARM [36]. We normalize the number of processing units to 16, the same as the implementation in CHARM. As shown in Table VI, we normalize one URAM as eight BRAM and report the total number of RAM used in both the designs. For the Softmax layer, since we replace the resource-demanding operations, i.e., exponential and division, we saved the number of DSP and LUT by 7.0 and 14.48×, respectively. Instead of using the double buffer technique applied in CHARM [36], by using the streaming pipelined architecture within this kernel, we save the LUTMem by 18 834× and total RAM by 9.19×. For the GeLU kernel, with the LUT optimization, it no longer consumes LUTMem, RAM, and DSP and reduces REG and LUT by 137 and 142×. We show the overall implementation layout of DeiT-T in Fig. 10 containing ten MM units and non-MM modules, including AXI DMA, Transpose, and nonlinear kernels.

⑤*Can We Leverage EQ-ViT When Model Sizes Do Not Fit On-Chip?* If a model can not fit on a single board, we can

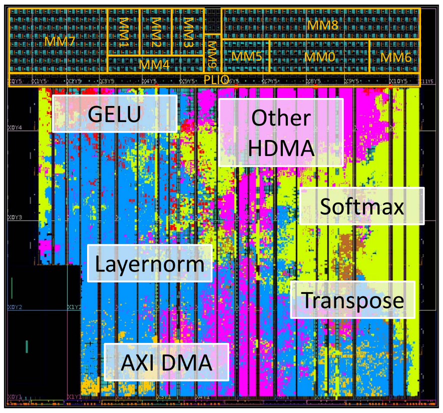

Fig. 10. EQ-ViT implementation layout on VCK190 with kernels highlighted in the FPGA and AIE portion of ACAP.

TABLE VII
COMPARISON OF THE TOP-1 (%) ACCURACY WITH SOTA METHODS ON MULTIPLE DATASETS

Model	FP32	PTQ					QAT		
		MinMax	EMA	Percentile	OMSE	FQ-ViT	LSQ	Q-ViT*	EQ-ViT
ImageNet Dataset									
DeiT-T	72.2	70.9	71.2	71.5	71.3	71.6	71.5	73.6	**74.5**
DeiT-160	68.1	67	67.6	67.8	67.9	68	67.9	70.1	**70.5**
DeiT-256	77.2	72.5	72.5	74	72.4	76.6	75.9	77.6	**78.2**
LV-ViT-T	79.1	75.4	75.4	76.9	75.3	77.4	78.7	80.1	**80.5**
Cifar-100 Dataset									
DeiT-T	85.6	85	85.1	85.3	85.1	85.4	85.3	86.2	**86.6**
DeiT-160	83.5	83	83.3	83.3	83.4	83.5	83.5	84.4	**84.4**
DeiT-256	87.1	85.8	85.9	86.5	85.7	87	86.9	88	**88.3**
LV-ViT-T	88.1	87.3	87.4	87.5	87.2	88.1	88.4	89.2	**89.5**
Cifar-10 Dataset									
DeiT-T	97.8	97.5	97.6	97.5	97.4	97.8	97.7	98.1	**98.3**
DeiT-160	96.3	96.1	96.2	96.3	96.1	96.4	96.5	96.9	**96.9**
DeiT-256	98.1	98	98	98.3	97.9	98.1	98	98.7	**98.9**
LV-ViT-T	98.7	98.6	98.6	98.7	98.5	98.8	98.6	99.2	**99.4**

Note: * indicates our reproduced results with quantized nonlinear operations for a fair comparison; And all the models (except FP32) are quantized into 8-bit precision.

leverage EQ-ViT to explore how the model is most effectively partitioned onto the multiple devices, which is our future work.

C. Inference Accuracy Comparisons

We compare EQ-ViT accuracy with the popular PTQ methods [59], [72], [73] and the SOTA QAT methods [51], [74]. For the sake of fairness, we reproduced the results of Q-ViT with quantized GeLU and Softmax.

Image Classification on Multiple Datasets: ① ImageNet. Recent SOTA methods for PTQ suffer a significant drop in accuracy up to 3.8% (Table VII). In contrast, ours can enhance the task accuracy up to 2.4% over the baseline by minimizing the quantization errors and removing the model redundancy. While the SOTA QAT method, Q-ViT, has made strides in correcting information distribution within ViT models, it still relies on the floating-point computations for Softmax and GeLU, making it challenging for the practical and efficient hardware deployment. In contrast, EQ-ViT leverages activation flow fitting and optimization to achieve an additional accuracy boost of 0.4%$\sim$0.9% over Q-ViT. Furthermore, EQ-ViT supports efficient implementation on ACAP. ② *Cifar-100 and Cifar-10.* We extend results on the Cifar datasets to showcase our validation. For the Cifar-100 dataset, EQ-ViT can enhance accuracy up to 1.4% and achieve 0.3% $\sim$ 0.4% higher accuracy than Q-ViT. For the Cifar-10 dataset, EQ-ViT can enhance accuracy up to 0.8%, and reach 0.2% higher accuracy than Q-ViT. Q-ViT introduces DGD distillation to distill the knowledge from the larger-size ViT to the smaller-size one, which is integrated into our training setting. Notably, EQ-ViT also surpasses the Q-ViT accuracy under the same training conditions.

VII. GENERALITY DISCUSSION AND MICROARCHITECTURE INSIGHTS

EQ-ViT performance improvements over the prior solutions come from two folds as follows.

1) *Software Aspect:* EQ-ViT accelerator mapping and optimization techniques that fully leverage all the heterogeneous microarchitecture features on ACAP. For those, we explain how different optimization techniques included in EQ-ViT contribute to the performance improvements and discuss whether and how those optimizations can be applied on the other platforms, including FPGA and GPU.
2) *Hardware Aspect:* The heterogeneous microarchitecture features from ACAP that provide flexible mapping features to be applied on such architecture. Specifically, those EQ-ViT mapping features that can not be ported to FPGAs or GPUs reflect the corresponding architecture limitations on FPGAs or GPUs.

Quantization: The performance gain from quantization comes from two parts: 1) the improved peak computation throughput and 2) the reduced off-chip data access. Especially, if the model size after quantization gets across a threshold and the weights can fit on-chip, there will be a huge improvement since all the intermediate data can be forwarded on-chip.

Accelerators on FPGA and ACAP can fully benefit from quantization, whereas GPU can not. Current GPU frameworks, e.g., TensorRT, can not fully cache intermediate data across different kernel function calls unless the users explicitly rewrite multiple kernels into one kernel (fusion). Another GPU software limitation is the implicit quantized kernels. In our GPU profiling for quantized models, TensorRT generates a mixed precision model, where the BMM kernels are computed in FP32 and not in IN8. If we can quantize the BMM, Softmax, LayerNorm, and transpose kernels in GPU, the hypothetical latency of DeiT-T on A10G GPU can be reduced to 1.05 ms, which is 1.9$\times$ when compared to the EQ-ViT latency.

On-Chip Forwarding: By applying on-chip forwarding, activations of the models can be kept inside the accelerator chip to reduce the off-chip communication. This technique has been applied to the Versal ACAP and FPGA platforms. On ACAP, applying this technique gives 3$\times$ latency reduction.

For GPU, the on-chip forwarding is limited compared to FPGA or ACAP. The flexibility in PL logic in FPGA and ACAP allows multiple accelerators to communicate with each other with arbitrary data forwarding per the user's control. In GPU, shared memory can be explicitly controlled by the user. However, one shared memory in one stream multiprocessor (SM) can not directly forward the data to the other shared

TABLE VIII
COMPARISONS OF FPGA, GPU, AND ACAP WITH SOTA FRAMEWORK IMPLEMENTATIONS (IMPL.) AND EQ-VIT OPTIMIZATIONS

Mapping features	FPGA+SOTA Impl. (HeatViT)	FPGA+EQ-ViT Optimizations	GPU+ SOTA Impl. (TensorRT)	GPU+EQ-ViT Optimizations	ACAP+SOTA Impl. (CHARM)	ACAP+EQ-ViT Optimization
Quantization	yes	yes	partial	partial ->yes	no	yes (4.2x)
On-Chip Forwarding	no	yes	no	arch limit	no	yes (3.4x)
Multi Spatial Accelerators	no	yes	no	arch limit	yes	yes (2.3x)
Fine-grained Pipelining	no	yes	no	arch limit	no	yes (2.7x)
Utilize AI-optimized PEs	no	arch limit	yes	yes	yes	yes
Estimated latency after EQ-ViT	7.3ms	3.9ms	1.8ms	1.05ms	50ms (1x)	0.561ms (89x)

memory in another SM. It has to go through the off-chip DDR or HBM. This is the microarchitecture limitation on GPU.[2]

Multiple Spatial Accelerators: On FPGA and ACAP platforms, compared with sequentially called one unified accelerator, the spatially called multiple accelerators can reach higher hardware utilization as each hardware accelerator has smaller hardware resources and can be specialized for the kernel.

In GPUs, horizontal fusion [76], [77] is motivated by similar reasons, i.e., using multiple kernels running at the same time instead of launching kernels sequentially. The key idea is to allocate different groups of SM working simultaneously whereas each SM group works on one type of the kernel. However, such multiple spatial accelerators in GPU have less flexibility than in FPGA and ACAP. The partition in GPU is in the SM granularity, therefore, different hardware resources, i.e., computation processing elements (PEs), and on-chip storage across different accs have a fixed ratio. In FPGA and ACAP, PL provides users with full flexibility to partition computation PE (DSPs, LUT, and AIEs) and on-chip storage (BRAM and URAM) with arbitrary ratios across different accs.

Fine-Grained Pipelining: Applying the fine-grained pipelining enables execution overlap among the accelerators, and leads to higher resource utilization and lower latency. Fine-grained pipelining can be easily implemented in FPGA and ACAP, on the contrary, it is not easily implemented on GPUs. We analyse the DeiT-T inference on A10G, if we can hack all the BMM kernels to be computed in INT8, the latency reduces from 1.8 to 1.05 ms, however, this can not be further reduced. The 1.05 ms latency includes MM kernels at 0.78 ms and non-MM kernels at 0.27 ms. Unlike ACAP, which allows full programmability and flexibility to allow AIE and FPGA within the ACAP SoC to run simultaneously, the current GPU programming model does not allow the simultaneous execution between the GPU Tensor cores and GPU CUDA cores.

VIII. SUMMARY AND CONCLUSION

We summarize our generality discussion in Table VIII. The FPGA platforms are highly flexible and support most of the EQ-ViT optimization methods. Without the AI-optimized PE like tensor cores or AI engine, the computing capability limits the performance of FPGAs. GPUs have the highest theoretical throughput and bandwidth, but the relatively fixed architecture limits their performance in latency-critical situations. The ACAP platform has both the flexibility and AI-optimized PE, thus reaching the lowest latency with the optimization of EQ-ViT.

This implies interesting research questions, e.g., what other kinds of applications will let ACAP, a combination of FPGA and AI-optimized SoC achieve the better of both the worlds? Shall we introduce FPGA or reconfigurable architecture in broader GPU architecture to improve the latency? If FPGA is too fine grained, what is the least reconfigurability needed in the future architecture to balance the performance and adaptability? We leave these in our future work.

[2]On-chip forwarding between SMs can not be implemented on Nvidia GPUs before ampere generation. However, as the successor of ampere architecture, the Hopper architecture uses distributed shared memory (DSMEM) [75], enabling fast communication between the shared memory and potentially providing more flexibility in on-chip forwarding among SMs on GPUs.

REFERENCES

[1] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021, pp. 1–22.
[2] H. Touvron et al., "Training data-efficient image transformers & distillation through attention," in *Proc. 38th ICML*, 2021, pp. 10347–10357.
[3] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. ICCV*, 2021, pp. 1–14.
[4] N. Carion et al., "End-to-end object detection with transformers," in *Proc. ECCV*, 2020, pp. 213–229.
[5] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," in *Proc. ICLR*, 2020, pp. 1–16.
[6] H. Chen et al., "Pre-trained image processing transformer," in *Proc. CVPR*, 2021, pp. 12299–12310.
[7] L. Zhou et al., "End-to-end dense video captioning with masked transformer," in *CVPR*, 2018, pp. 8739–8748.
[8] W. Wang et al., "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. ICCV*, 2021, pp. 548–558.
[9] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," in *Proc. NeurIPS*, 2021, pp. 1–14.
[10] H. Cao et al., "Swin-unet: Unet-like pure transformer for medical image segmentation," 2021, *arXiv:2105.05537*.
[11] M. Rafie. "Autonomous vehicles drive AI advances for edge computing." 2021. [Online]. Available: https://www.3dincites.com/2021/07/autonomous-vehicles-drive-ai-advances-for-edge-computing/
[12] (CERN, Geneva, Switzerland). *Colliding Particles Not Cars: CERN'S Machine Learning Could Help Self-Driving Cars*. (2023). Accessed: Jan. 25, 2023. [Online]. Available: https://home.cern/news/news/knowledge-sharing/colliding-particles-not-cars-cerns-machine-learning-could-help-self
[13] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai, and D. Bharadia, "EdgeRIC: Empowering realtime intelligent optimization and control in nextG networks," 2023, *arXiv:2304.11199*.
[14] G. Feng et al., "RisGraph: A real-time streaming system for evolving graphs to support sub-millisecond per-update analysis at millions ops/s," in *Proc. SIGMOD*, 2021, pp. 513–527.
[15] Y. Li et al., "EfficientFormer: Vision transformers at mobileNet speed," in *Proc. 36th Adv. Neural Inf. Process. Syst.*, 2022, pp. 1–16.
[16] Y. Li et al., "Rethinking vision transformers for mobilenet size and speed," 2022, *arXiv:2212.08059*.

[17] Y. Nagamatsu, F. Sugai, K. Okada, and M. Inaba, "Basic implementation of FPGA-GPU dual SOC hybrid architecture for low-latency multi-DOF robot motion control," in *Proc. IROS*, 2020, pp. 7255–7260.

[18] D. Gizopoulos et al., "Architectures for online error detection and recovery in multicore processors," in *Proc. DATE*, 2011, pp. 1–6.

[19] R. Basir et al., "Fog computing enabling industrial Internet of Things: State-of-the-art and research challenges," *Sensors*, vol. 19, no. 21, p. 4807, 2019.

[20] P. Koppermann, F. De Santis, J. Heyszl, and G. Sigl, "Low-latency x25519 hardware implementation: Breaking the 100 microseconds barrier," *Microprocess. Microsyst.*, vol. 52, pp. 491–497, Jul. 2017.

[21] J. Soifer et al., "Deep learning inference service at microsoft," in *Proc. OpML*, 2019, pp. 1–4.

[22] A. Putnam et al., "A reconfigurable fabric for accelerating large-scale datacenter services," *IEEE Micro*, vol. 35, no. 3, pp. 10–22, May/Jun. 2015.

[23] J. Fowers et al., "A configurable cloud-scale DNN processor for real-time AI," in *Proc. ISCA*, 2018, pp. 1–14.

[24] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th ISCA*, 2017, pp. 1–12.

[25] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, May/Jun. 2018.

[26] N. Jouppi et al., "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proc. ISCA*, 2023, pp. 1–14.

[27] F. G. de Magalhães et al., "Optical interconnection networks: The need for low-latency controllers," in *Photonic Interconnects for Computing Systems*. Aalborg, Denmark: River Publ., 2022, pp. 73–105.

[28] M. Miscuglio and V. J. Sorger, "Photonic tensor cores for machine learning," *Appl. Phys. Rev.*, vol. 7, no. 3, Jul. 2020, Art. no. 031404.

[29] F. P. Sunny, E. Taheri, M. Nikdast, and S. Pasricha, "A survey on silicon photonics for deep learning," *J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 4, pp. 1–57, Jun. 2021.

[30] P. Dong et al., "HeatViT: Hardware-efficient adaptive token pruning for vision transformers," in *Proc. HPCA*, 2023, pp. 442–455.

[31] H. Vanholder, "Efficient inference with tensorRT," in *Proc. GPU Technol. Conf.*, vol. 1, 2016, pp. 1–24.

[32] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020, *arXiv:2004.09602*.

[33] "Nvidia Nsight systems." NVIDIA Developer. Accessed: Aug. 22, 2024. [Online]. Available: https://developer.nvidia.com/nsight-systems

[34] "Nvidia TensorRT documentation." Nvidia. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html#working-with-int8

[35] J. Zhuang, Z. Yang, and P. Zhou, "High performance, low power matrix multiply design on ACAP: From architecture, design challenges and DSE perspectives," in *Proc. DAC*, 2023, pp. 1–6.

[36] J. Zhuang et al., "CHARM: Composing heterogeneous accelerators for matrix multiply on versal ACAP architecture," in *Proc. FPGA*, 2023, pp. 153–164.

[37] H. Kwon et al., "Heterogeneous dataflow accelerators for multi-DNN workloads," in *proc. HPCA*, 2021, pp. 71–83.

[38] S.-C. Kao and T. Krishna, "MAGMA: An optimization framework for mapping multiple DNNs on multiple accelerator cores," in *Proc. HPCA*, 2022, pp. 814–830.

[39] T. Wang et al., "ViA: A novel vision-transformer accelerator based on FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4088–4099, 2022.

[40] H. You et al., "ViTCoD: Vision transformer acceleration via dedicated algorithm and accelerator co-design," in *Proc. HPCA*, 2023, pp. 1–14.

[41] Z. Lit et al., "Auto-ViT-acc: An FPGA-aware automatic acceleration framework for vision transformer with mixed-scheme quantization," in *Proc. FPL*, 2022, pp. 109–116.

[42] J. Zhuang et al., "SSR: Spatial sequential hybrid architecture for latency throughput tradeoff in transformer acceleration," in *Proc. FPGA*, 2024, pp. 55–66.

[43] Z. Zong, G. Song, and Y. Liu, "DETRs with collaborative hybrid assignments training," in *Proc. ICCVW*, 2023, pp. 6748–6758.

[44] W. Lv et al., "DETRs beat YOLOs on real-time object detection," 2023, *arXiv:2304.08069*.

[45] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 1–15.

[46] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "HAWQ: Hessian aware quantization of neural networks with mixed-precision," in *Proc. ICCVW*, 2019, pp. 293–302.

[47] Y. Li et al., "BRECQ: Pushing the limit of post-training quantization by block reconstruction," 2021, *arXiv:2102.05426*.

[48] A. H. Zadeh et al., "GOBO: Quantizing attention-based NLP models for low latency and energy efficient inference," in *Proc. 53rd MICRO*, 2020, pp. 811–824.

[49] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," *Proc. NeurIPS*, vol. 33, 2021, pp. 28089–28154.

[50] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers," in *Proc. NeurIPS*, vol. 35, 2022, pp. 1–24.

[51] Y. Li et al., "Q-ViT: Accurate and fully quantized low-bit vision transformer," in *Proc. NeurIPS*, vol. 34, 2022, pp. 28092–28103.

[52] T. Chen et al., "Chasing sparsity in vision transformers: An end-to-end exploration," in *Proc. NeurIPS*, vol. 34, 2021, pp. 19974–19988.

[53] M. Zhu, Y. Tang, and K. Han, "Vision transformer pruning," 2021, *arXiv:2104.08500*.

[54] S. Yu et al., "Unified visual transformer compression," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–17.

[55] S.-C. Kao, S. Subramanian, G. Agrawal, A. Yazdanbakhsh, and T. Krishna, "FLAT: An optimized dataflow for mitigating attention bottlenecks," in *Proc. ASPLOS*, 2023, pp. 1–17.

[56] X. Zhang et al., "DNNExplorer: A framework for modeling and exploring a novel paradigm of FPGA-based DNN accelerator," in *Proc. ICCAD*, 2020, pp. 1–9.

[57] M. S. Bensaleh et al., "Optimal task scheduling for distributed cluster with active storage devices and accelerated nodes," *IEEE Access*, vol. 6, pp. 48195–48209, 2018.

[58] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[59] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. CVPR*, 2018, pp. 2704–2713.

[60] J. Cai, M. Takemoto, and H. Nakajo, "A deep look into logarithmic quantization of model parameters in neural networks," in *Proc. 10th IAIT*, 2018, pp. 1–8.

[61] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, "FQ-ViT: Post-training quantization for fully quantized vision transformer," in *Proc. IJCAI*, 2022, pp. 1173–1179.

[62] W. contributors. "68–95–99.7 rule." Wikipedia. 2022. [Online]. Available: https://en.wikipedia.org/wiki/68

[63] G. C. Cardarilli et al., "A pseudo-softmax function for hardware-based high speed image classification," *Sci. Rep.*, vol. 11, Jul. 2021, Art. no. 15307.

[64] W. Wang, S. Zhou, W. Sun, P. Sun, and Y. Liu, "SOLE: Hardware-software co-design of softmax and layerNorm for efficient transformer inference," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2023, pp. 1–9.

[65] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-BERT: Integer-only bert quantization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5506–5518.

[66] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009, pp. 248–255.

[67] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. TR-2009, 2009.

[68] Z. Jiang et al., "All tokens matter: Token labeling for training better vision transformers," 2021, *arXiv:2104.10858*.

[69] "System management interface SMI | NVIDIA developer," Nvidia. [Online]. Available: https://developer.nvidia.com/nvidia-system-management-interface

[70] "ViT Github." Meta. [Online]. Available: https://github.com/facebookresearch/deit

[71] *Board Evaluation and Management Tool*, AMD/Xilinx, San Jose, CA, USA, document UG1573, 2023.

[72] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, and R. Fan, "Fully quantized network for object detection," in *Proc. CVPR*, 2019, pp. 2810–2819.

[73] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," in *Proc. ICCVW*, 2019, pp. 3009–3018.

[74] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*.

[75] "NVIDIA H100 tensor core GPU architecture." Nvidia. [Online]. Available: https://resources.nvidia.com/en-us-tensor-core

[76] L. Ma et al., "Rammer: Enabling holistic deep learning compiler optimizations with {rTasks}," in *Proc. OSDI*, 2020, pp. 881–897.

[77] A. Li, B. Zheng, G. Pekhimenko, and F. Long, "Automatic horizontal fusion for GPU kernels," in *Proc. CGO*, 2022, pp. 14–27.