

# An FPGA-Based Reconfigurable Accelerator for Convolution-Transformer Hybrid EfficientViT

Haikuo Shao<sup>1</sup>, Huihong Shi<sup>1</sup>, Wendong Mao<sup>2</sup>, and Zhongfeng Wang<sup>1,2</sup>

<sup>1</sup>School of Electronic Science and Engineering, Nanjing University, Nanjing, China

<sup>2</sup>School of Integrated Circuits, Sun Yat-sen University, Shenzhen, China

Email: {hkshao, shihh}@smail.nju.edu.cn, maowd@mail.sysu.edu.cn, zfwang@nju.edu.cn

**Abstract**—Vision Transformers (ViTs) have achieved significant success in computer vision. However, their intensive computations and massive memory footprint challenge ViTs’ deployment on embedded devices, calling for efficient ViTs. Among them, EfficientViT, the state-of-the-art one, features a Convolution-Transformer hybrid architecture, enhancing both accuracy and hardware efficiency. Unfortunately, existing accelerators cannot fully exploit the hardware benefits of EfficientViT due to its unique architecture. In this paper, we propose an FPGA-based accelerator for EfficientViT to advance the hardware efficiency frontier of ViTs. Specifically, we design a reconfigurable architecture to efficiently support various operation types, including lightweight convolutions and attention, boosting hardware utilization. Additionally, we present a time-multiplexed and pipelined dataflow to facilitate both intra- and inter-layer fusions, reducing off-chip data access costs. Experimental results show that our accelerator achieves up to 780.2 GOPS in throughput and 105.1 GOPS/W in energy efficiency at 200MHz on the Xilinx ZCU102 FPGA, which significantly outperforms prior works.

**Index Terms**—Vision Transformer, convolution, hybrid architecture, hardware accelerator, FPGA

## I. INTRODUCTION

Recently, Vision Transformers (ViTs) have been proposed and attracted increasing attention in the computer vision field [1], [2]. Despite ViTs’ remarkable performance against their convolution-based counterparts, the intensive computations and huge memory footprint during inference pose challenges to ViTs’ deployment on resource-constrained devices [3], [4]. Particularly, the computational complexity of the self-attention mechanism in standard ViTs is quadratic w.r.t. the number of input tokens, limiting ViTs’ real-world application on high-resolution images. Besides, the non-linear operations in ViTs, e.g., LayerNorm (LN), GELU [5], and especially Softmax, are hardware unfriendly and quantization sensitive [6], [7], hindering ViTs’ achievable task accuracy and hardware efficiency.

To promote ViTs’ deployment, various efforts have been devoted to the development of efficient ViTs [8]–[11], which replace the vanilla computational-intensive self-attention in standard ViTs with more efficient alternatives that exhibit linear computational complexity. However, it has been widely demonstrated that the simplification of the attention mechanism inevitably results in a reduction in local feature extraction capabilities. This limitation necessitates the incorporation of

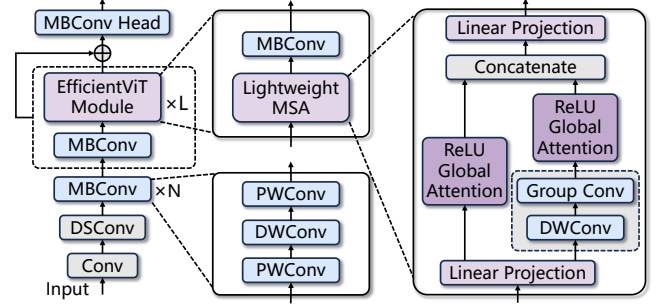


Fig. 1. The macro architecture of EfficientViT [8]. Each MBConv consists of two pointwise convolutions (PWConvs) separated by a depthwise convolution (DWConv). Besides, the key component of the EfficientViT module is the lightweight Multi-Scale Attention (MSA).

supplementary components such as convolutions [8], [11], yielding hybrid architectures for efficient ViTs that integrate both convolutions and Transformer blocks. Particularly, the state-of-the-art (SOTA) efficient ViT, dubbed EfficientViT [8], can achieve higher accuracy than Swin-T [12] (by +1.4%) and DeiT [2] (by +2.9%) with a comparable number of parameters. As illustrated in Fig. 1, EfficientViT features a Convolution-Transformer hybrid architecture, primarily comprising MBConvs [13] and EfficientViT Modules. The latter includes a Softmax-free and lightweight Multi-Scale Attention (MSA), aiming to enhance both hardware efficiency and representation capability. Besides, EfficientViT also replaces vanilla LN and GELU in standard ViTs with hardware-friendly BatchNorm (BN) and Hardswish [14], respectively.

Despite EfficientViT’s effectiveness, existing accelerators [3], [15]–[17] are mainly dedicated to standard ViTs [1], [2] and not directly applicable to accelerate EfficientViT. To fully unleash its hardware benefit potential, it is highly desired to develop a dedicated accelerator for EfficientViT, which, however, poses challenges due to its dynamic workloads and high-intensity memory access demands. Particularly, EfficientViT involves various operation types, including lightweight convolutions (i.e., MBConvs) with different kernel sizes, strides, and feature dimensions, as well as the lightweight attention (i.e., MSA), which exhibits distinct computational patterns compared to the vanilla self-attention in standard ViTs. Moreover, the aforementioned lightweight components in EfficientViT exhibit reduced computing parallelism and fewer data reuse opportunities than their standard counterparts, yielding either high memory bandwidth requirement or low computation

This work was supported in part by the National Key R&D Program of China under Grant 2022YFB4400604. (Corresponding Author: Wendong Mao and Zhongfeng Wang)

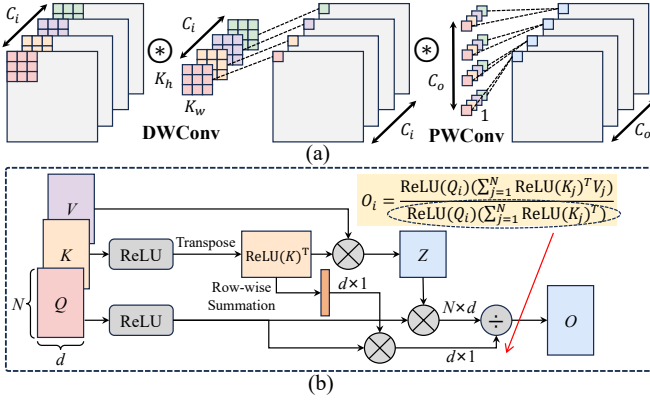


Fig. 2. (a) DSConv: Depthwise Convolution followed by Pointwise Convolution. (b) The computation flow of ReLU-based global attention in EfficientViT.

resource utilization. Thus, in this paper, we present an FPGA-based accelerator for EfficientViT to tackle these challenges. The main contributions are summarized as follows.

- A reconfigurable architecture is designed to efficiently support various operation types in the Convolution-Transformer hybrid architecture of EfficientViT, including lightweight convolutions and lightweight attention.
- A novel time-multiplexed and pipelined dataflow is proposed to fuse computations among adjacent lightweight convolutions and computations within lightweight attention, dramatically boosting computing resource utilization while easing bandwidth requirements.
- Based on optimizations of both computation and communication, an accelerator dedicated to EfficientViT is developed. It is implemented on the Xilinx ZCU102 FPGA platform at 200MHz and achieves up to 780.2 GOPS in throughput and 105.1 GOPS/W in energy efficiency.

## II. STRUCTURE OF EFFICIENTViT

As depicted in Fig. 1, EfficientViT [8] has an input stem of a generic convolution (Conv) followed by a DSConv layer, which is a combination of a depth-wise convolution (DWConv in Fig. 2(a) left) and a point-wise convolution (PWConv in Fig. 2(a) right). After that, two key types of blocks are involved in EfficientViT: the **MBConv** [13] and the **EfficientViT Module**. The **MBConv** features two PWConvs separated by a DWConv. Each layer is followed by BatchNorm (BN) and Hardswish activation [14] (except the final PWConv). Notably, BN can be implemented via  $1 \times 1$  convolutions, which can be integrated into preceding convolutions to facilitate quantization and acceleration [18]. In addition, each **EfficientViT Module** comprises a lightweight Multi-Scale Attention (MSA) and an MBConv to separately extract context and local information. In the MSA, inputs are projected to produce query/key/value ( $Q/K/V$ ). Then, they are processed by lightweight convolutions to obtain multi-scale tokens, which further undergo ReLU-based global attention. Finally, the results are concatenated and projected to generate the final outputs. As illustrated in Fig. 2(b), the ReLU-based global attention transforms the similarity function in the original Softmax-based attention (i.e.,  $\text{Exp}(QK^T/\sqrt{d})$ ) into  $\text{ReLU}(Q)\text{ReLU}(K)^T$ , thus not only

eliminating the need for Softmax but also achieving linear computational complexity by utilizing the associative property of matrix multiplication.

## III. PROPOSED HARDWARE DESIGN

As discussed above, there are four main types of operations in the backbone of EfficientViT: generic Convs, PWConvs, DWConvs, and matrix multiplications (MatMuls). As MatMuls can be treated as PWConvs with large batch sizes, an efficient hardware architecture that can effectively handle MSA and various types of convolutions is highly desired. Additionally, lightweight operations (i.e., PWConvs/DWConvs/MSA) in EfficientViT features reduced computing parallelism and fewer data reuse opportunities, calling for an effective dataflow to enhance hardware utilization and ease bandwidth requirements.

### A. Multipliers and Adder-Trees Design Paradigm

Convolutions in neural networks can be fundamentally decomposed into a series of multiplication and addition computations. Hence, parallelized hardware architectures incorporating multipliers and adder-trees (MAT) offer a straightforward and efficient solution for generic Convs and PWConvs (which are essentially generic Convs with  $1 \times 1$  kernels). Specifically, inputs or weights can be broadcast to all MATs to facilitate data reuse. Within each MAT, multiple multipliers are responsible for generating partial sums along the input channel dimension in parallel, which are then added (via the adder tree) and accumulated to obtain the final output.

Despite its effectiveness, MAT-based architecture cannot effectively support DWConvs in EfficientViT due to its fixed structure and dataflow. Particularly, DWConvs handle each input channel separately, thus only partial sums within the the same sliding window can be summed and accumulated. This constrains the achievable parallelism within each MAT to DW's kernel size, limiting MAT's flexibility for supporting various kernel sizes as well as its scalability to a large scale. Moreover, DWConvs with different kernel sizes and strides yield distinct overlap patterns between adjacent sliding windows when conducting convolutions, resulting in significant buffer overheads or complex memory management to support the generation of consecutive output pixels within MATs [19].

### B. Reconfigurable Architecture Design

To boost flexibility, we develop a reconfigurable processing element (RPE) architecture to efficiently support various types of convolutions in EfficientViT. As depicted in Fig. 3 (b), RPE contains  $M$  PE lines, each has  $N$  multiplication-accumulation (MAC) units in parallel. It can be reconfigured to operate in both DW mode and PW mode to support DWConvs and PWConvs/generic Convs, respectively:

1) *DW Mode*: When RPE works in the DW Mode, partial sums within each sliding window are accumulated within each MAC, which is an **self-accumulation** dataflow. Fig. 3(a) shows an example of executing of a  $k \times k$  ( $k = 3$  here) DWConv with stride = 1 on the RPE with  $M = 4$ . In the first cycle, inputs  $a_1$  to  $a_M$  are read and individually transferred

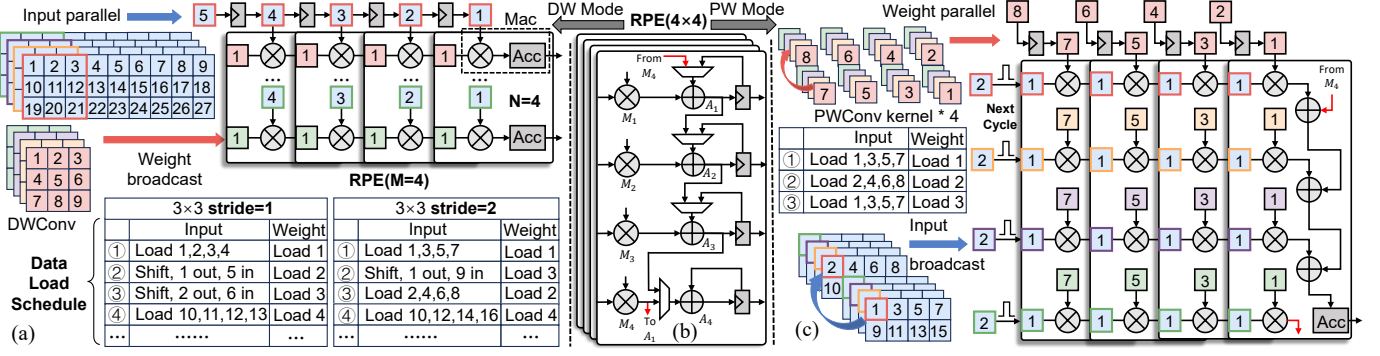


Fig. 3. (a) RPE works in DW Mode. (b) The micro-architecture of proposed Reconfigurable Processing Element (RPE). (c) RPE works in PW Mode.

to top MACs of different PE lines, with weight  $w_1$  loaded and broadcast. In the second clock cycle, input data are right-shifted along registers with  $a_1$  dequeued and  $a_{M+1}$  enqueued. Weight is updated to  $w_2$ . Next cycle processes  $a_3 \sim a_{M+2}$  and  $w_3$ . After  $k$  cycles, the computation moves to the next row of the input feature map, following the same pattern. When  $k$  rows of data are processed, the output  $o_1$  to  $o_M$  for DWConv can be obtained from the top MACs of  $M$  PE lines via self-accumulation. The  $N$  MACs within each PE line can conduct computations for  $N$  output channels in parallel.

For  $k \times k$  DWConv with a stride of 2, overlaps among consecutive sliding windows are spaced instead of successive. Thus, odd-column-indexed pixels within each row of the input feature map are first read and right-shifted by cycles, followed by the even-column-indexed pixels. Weights are also broadcast following the same “first odd, then even” order to accommodate this modified computation scheme.

2) *PW Mode*: When RPE works in the PW mode to support both PWConvs and generic Convs, it performs a similar functionality as the MAT-based architecture. Specifically, as shown in Fig. 3 (c), partial sums along the input channel are computed via  $N$  multipliers within each PE line and then accumulated down-forward the PE line, which is **down-forward accumulation** dataflow. This implies that the parallelism within each PE line is along the input channel dimension to leverage the partial sum reuse opportunity. Besides, inputs can be broadcast among  $M$  PE lines to exploit input reuse.

### C. Overall Architecture of Proposed Accelerator

Considering MAT’s efficiency in executing the dominant PWConvs in EfficientViT and RPE’s flexibility in supporting various operation types, we propose our architecture in Fig. 4 incorporating both components to marry the best of both designs. Particularly, our accelerator mainly comprises multiple on-chip buffers and  $L$  parallel processing groups (PGs), each containing an RPE engine and a MAT engine. The RPE engine can flexibly process DWConvs, PWConvs, generic Convs, and MatMuls, while the MAT engine is responsible for efficiently executing the latter three.

As illustrated in Fig. 4, buffers A, B, and C are used to cache various types of data, including weight  $W$ , input  $A$ , as well as query  $Q$ , key  $K$ , and value  $V$  within MSA. During computation, data from buffers A and C are transmitted to all

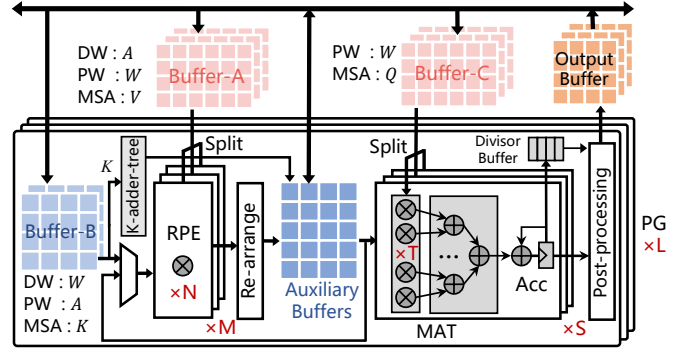


Fig. 4. The overall architecture of our accelerator. The RPE engine is composed of  $M$  PE lines with  $M \times N$  multipliers, and the MAT engine is configured as  $S$  MATs with  $S \times T$  multipliers.

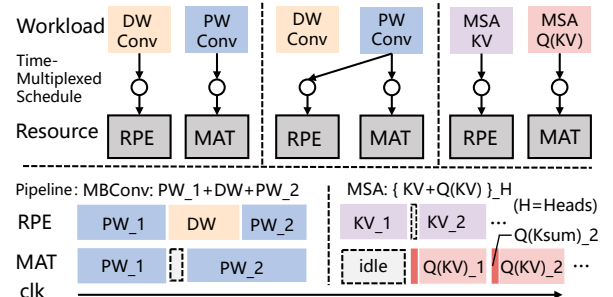


Fig. 5. The proposed time-multiplexed and pipelined (TMP) dataflow.

PGs. Then, they are split and separately sent to  $M$  PE lines and  $S$  MAT lines within each PG. Besides, data read from the internal buffer B can be broadcast to  $M$  PE lines. The auxiliary buffers can connect the RPE and MAT engines and can also directly communicate with the off-chip DRAM.

Additionally, as shown in Fig. 2 (b), in addition to MatMuls, MSA also involves row-wise summations and divisions, thus our accelerator also integrates auxiliary K-adder-tree and dividers to accommodate MSA’s computation.

### D. Time-Multiplexed and Pipelined Dataflow

To enhance PE utilization and reduce data costs, we further equip our accelerator with a time-multiplexed and pipelined (TMP) dataflow to facilitate both the (1) **inter-layer** fusion in MBConvs and (2) **intra-layer** fusion for computations within MSA. Firstly, considering that DWConvs can only be executed on the RPE engine and they are always followed by PWConvs in MBConvs of EfficientViT, we thus fuse



TABLE I  
FPGA RESOURCE UTILIZATION

	LUT	FF	BRAM	DSP
Used	104463	249473	160	1024
Available	274080	548160	912	2520
Utilization	38.11%	45.51%	17.54%	40.63%

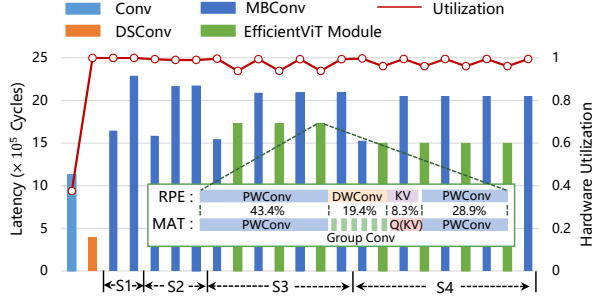


Fig. 6. The latency and hardware utilization evaluated on EfficientViT-B1, containing a generic Conv, a DSCConv layer, and four stages (S1-S4).

DWConvs with their subsequent PWConvs. As depicted in Fig. 5, when DWConv is executed on the RPE engine, partial sums can be cached in the auxiliary buffer, then generated outputs can be immediately passed to the idle MAT engine to serve as input for its subsequent PWConv. As DWConvs involve fewer computations than PWConvs, when the RPE engine finishes processing DWConv, it can join the computation of the concurrent PWConv to boost hardware utilization.

Additionally, as for intra-layer fusion for MSA, MatMuls of  $Z = \text{ReLU}(K^T) \cdot V$  and  $\text{ReLU}(Q) \cdot Z$  can also be pipeline-executed on two engines. Specifically, when  $\text{ReLU}(K^T)$  is loaded from Buffer  $B$  to the RPE engine for conducting MatMuls with  $V$ , K-adder-tree can simultaneously perform row-wise summations of  $\text{ReLU}(K^T)$  to obtain  $\text{ReLU}(K)_{\text{sum}}^T$ . The resultant outputs, i.e.,  $\text{ReLU}(K)_{\text{sum}}^T$  and  $Z$  are saved in the auxiliary buffers and then broadcast to the MAT engine to sequentially conduct multiplications with  $Q$ , generating divisors and dividends of MSA, respectively. During this process, the pre-generated divisors are temporarily saved in a small divisor buffer. Once dividends are computed by the MAT engine, they can be divided by the previously saved divisors via dividers in the post-processing module (Fig. 4) to accomplish the final divisions of MSA.

#### IV. EXPERIMENTAL RESULTS

##### A. Experimental Setup

Our accelerator is coded with Verilog, synthesized and implemented by Vivado Design Suite, and evaluated on Xilinx ZCU102 FPGA at 200-MHz frequency. The hardware resource of  $(M \times N + S \times T) \times L$  is configured as  $(8 \times 8 + 8 \times 8) \times 16$ . Each multiplier in both RPE and MAT engines can execute the  $8 \times 8$ -bit fixed-point (FIX8) multiplication. Thus, to enhance DSP utilization, we adopt the SOTA DSP packaging method [20] to accommodate two  $8 \times 8$ -bit multiplications within each DSP following Auto-ViT-Acc [17] for fair comparisons. The resource consumption is reported in Table I.

##### B. Performance Analyses

From Fig. 6, which is evaluated on the EfficientViT-B1 [8] model, we can draw the following conclusions: (1) As the input image has only 3 channels and thus cannot be effectively mapped to our accelerator with high parallelism, this results in a low hardware utilization of 37.5% when executing the first generic Conv on our design; (2) Similarly, the group Convs in MSA also have fewer input channel parallelism opportunities than PWConvs, yielding a slight utilization decrease here. (3) However, due to the effectiveness of our proposed TMP dataflow in fusing dominant computations among DWConvs and PWConvs as well as within MSA, the overall utilization is above 95%, achieving a throughput of 780.2 GOPS and demonstrating our superiority.

TABLE II  
COMPARISONS WITH SOTA WORKS

	Efficient ViT [8]	ViA [16]	Auto-ViT-Acc [17]	Our work
Device	CPU*	Xilinx Alveo U50	Xilinx ZCU102	Xilinx ZCU102
Frequency (GHz)	1.8-3.0	0.3	0.15	0.2
Precision	FP32	FP16	FIX8	FIX8
DSP Used	-	2420	1936	1024
Throughput (GOPS)	54.7	309.6	711.2	780.2
Power (W)	11	39	8.46	7.43
Energy Efficiency (GOPS/W)	4.97	7.92	84.1	105.1
DSP Efficiency (GOPS/DSP)	-	0.13	0.37	0.76

\* Qualcomm Snapdragon 8Gen1 CPU with 11W peak power consumption.

##### C. Comparisons and Discussion

To verify our accelerator when executing EfficientViT, we compare with prior works: EfficientViT-B1 [8] measured on a mobile CPU with FP32 format, a SOTA Swin-Transformer [12] (also an efficient ViT) accelerator ViA [16] with FP16 format, and a standard ViT (DeiT) accelerator Auto-ViT-Acc [17] with FIX8 precision. From Table II, we can see that: (1) Compared with EfficientViT on CPU, we can gain  $14.3\times$  speedup and  $21.1\times$  energy efficiency; (2) Compared to ViA, our design achieves  $2.0\times$  throughput,  $13.3\times$  energy efficiency, and  $5.9\times$  DSP efficiency; (3) Although Auto-ViT-Acc consumes  $1.9\times$  more DSP resources than us, we can offer  $1.1\times$  throughput,  $1.25\times$  energy efficiency, and  $2.1\times$  DSP efficiency, further validating our effectiveness.

#### V. CONCLUSION

In this paper, we proposed an FPGA-based accelerator for Convolution-Transformer hybrid networks like EfficientViT. Specifically, we design a reconfigurable design to effectively support various types of convolutions and the Multi-Scale Attention (MSA). Furthermore, we propose a time-multiplexed and pipelined dataflow to facilitate layer/computation fusions, boosting hardware utilization and minimizing bandwidth requirement. Implemented results show that we can achieve up to 780.2 GOPS in throughput and 105.1 GOPS/W in energy efficiency, significantly outperforming prior works.

## REFERENCES

- [1] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *ArXiv*, vol. abs/2010.11929, 2020.
- [2] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention,” in *International Conference on Machine Learning*, 2020.
- [3] H. You, Z. Sun, H. Shi, Z. Yu, Y. Zhao, Y. Zhang, C. Li, B. Li, and Y. Lin, “Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design,” *ArXiv*, vol. abs/2210.09573, 2022.
- [4] J. Dass, S. Wu, H. Shi, C. Li, Z. Ye, Z. Wang, and Y. Lin, “Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention,” *ArXiv*, vol. abs/2211.05109, 2022.
- [5] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” *arXiv: Learning*, 2016.
- [6] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, “Fq-vit: Post-training quantization for fully quantized vision transformer,” in *International Joint Conference on Artificial Intelligence*, 2021.
- [7] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, “Ptq4vit: Post-training quantization framework for vision transformers,” *ArXiv*, vol. abs/2111.12293, 2021.
- [8] H. Cai, C. Gan, and S. Han, “Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition,” *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [9] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 548–558, 2021.
- [10] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, “Pvt v2: Improved baselines with pyramid vision transformer,” *Computational Visual Media*, vol. 8, pp. 415 – 424, 2021.
- [11] D. Han, X. Pan, Y. Han, S. Song, and G. Huang, “Flatten transformer: Vision transformer using focused linear attention,” *ArXiv*, vol. abs/2308.00442, 2023.
- [12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.
- [13] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.
- [14] A. G. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for mobilenetv3,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.
- [15] M. Sun, H. Ma, G. Kang, Y. Jiang, T. Chen, X. Ma, Z. Wang, and Y. Wang, “Vaqf: Fully automatic software-hardware co-design framework for low-bit vision transformer,” *ArXiv*, vol. abs/2201.06618, 2022.
- [16] T. Wang, L. Gong, C. Wang, Y. Yang, Y. Gao, X. Zhou, and H. Chen, “Via: A novel vision-transformer accelerator based on fpga,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, pp. 4088–4099, 2022.
- [17] Z. Li, M. Sun, A. Lu, H. Ma, G. Yuan, Y. Xie, H. Tang, Y. Li, M. E. Leeser, Z. Wang, X. Lin, and Z. Fang, “Auto-vit-acc: An fpga-aware automatic acceleration framework for vision transformer with mixed-scheme quantization,” *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 109–116, 2022.
- [18] Y. Zhang, B. Sun, W. Jiang, Y. Ha, M. Hu, and W. Zhao, “Wsq-addernet: Efficient weight standardization based quantized addernet fpga accelerator design with high-density int8 dsp-lut co-packing optimization,” *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2022.
- [19] Y. Yu, T. Zhao, K. Wang, and L. He, “Light-opu: An fpga-based overlay processor for lightweight convolutional neural networks,” *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020.
- [20] Xilinx, “Wp486: Deep learning with int8 optimization on xilinx devices,” in *White Paper*, 2017.