ViT-ToGo : Vision Transformer Accelerator with Grouped Token Pruning

Seungju Lee, Kyumin Cho, Eunji Kwon, Sejin Park, Seojeong Kim, and Seokhyeong Kang*
Department of Electrical Engineering, POSTECH, Pohang, South Korea
*shkang@postech.ac.kr

***Abstract*—Vision Transformer (ViT) has gained prominence for its performance in various vision tasks but comes with considerable computational and memory demands, posing a challenge when deploying it on resource-constrained edge devices. To address this limitation, various token pruning methods have been proposed to reduce the computation. However, the majority of token pruning techniques do not account for practical use in actual embedded devices, which demand a significant reduction in computational load. In this paper, we introduce ViT-ToGo, a ViT accelerator with grouped token pruning. This enables the parallel execution of the ViT models and the token pruning process. We implement grouped token pruning with a head-wise importance estimator which simplifies the process need for token pruning, including sorting and reordering. Our proposed method achieves up to 66% reduction in the number of tokens, resulting in up to 36% reduction in GFLOPs, with only a minimal accuracy drop of around 1%. Furthermore, the hardware implementation incurs a marginal resource overhead of 1.13% in average.**

I. Introduction

The Transformer architecture has recently gained considerable attention in various natural language processing (NLP) tasks, and extensive research efforts have explored its applicability to vision tasks. Vision Transformer (ViT) [1], a pioneer in adapting the transformer architecture for vision tasks, has demonstrated promising performance across computer vision domains, including image classification, object detection, and semantic segmentation.

However, the computational and memory demands of ViT present challenges when deploying it on resource-constrained edge devices. Various compression techniques, such as dynamic token pruning [2]–[4] and static weight pruning [5], [6], as well as quantization [7], [8], have been explored to mitigate these challenges. As observed in Table I, it is evident that Vision Transformers are significantly influenced by the number of tokens, particularly demonstrating a quadratic impact in Multi-Head Self-Attention (MHSA) mechanism. The quadratic computational complexity of ViT concerning token counts can be effectively alleviated through the application of token pruning methods. Notably, token pruning, which reduces the number of tokens in ViT, is of particular interest due to ViT's quadratic computational complexity concerning token count.

This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT).(No. RS-2023-00222085, Development of memory module and memory compiler for non-volatile PIM) and was supported by nanomaterials development program through the National Research Foundation of Korea (NRF) (2022M3H4A1A04096496) funded by the Ministry of Science and ICT, Korea.

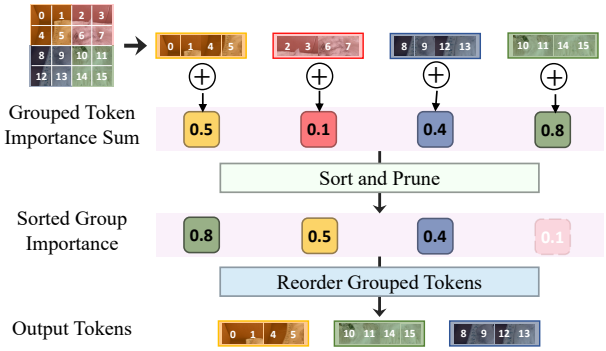


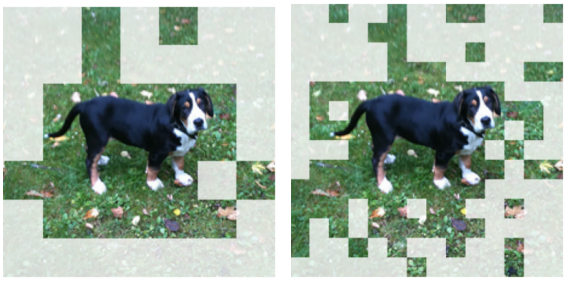
Fig. 1: Grouped Token Pruning Procedure. Important groups are retained and passed on to the next layer. Each group consists of tokens located adjacently in their respective image positions.

Existing token pruning approaches [2], [3], [9]–[13] have shortcomings in two critical aspects: they do not adequately consider the hardware footprint and latency implications for devices with limited computing resources, and their validation primarily takes place on conventional computing platforms like GPUs and CPUs, neglecting the parallelism capabilities of specialized hardware such as FPGAs.

In this paper, we introduce a hardware-accelerated token pruning algorithm for Vision Transformers (ViT), optimizing for hardware parallelism and minimizing latency and extra hardware resource. Our contributions are as follows:

- Grouped Token Pruning: Unlike conventional methods such as [2], [3], which simply sort and eliminate low-ranked individual tokens, we leverages the inherent similarities among adjacent image patches. We group these patches, then sort and prune them based on their collective importance (Fig.1). This approach is able to reduce additional cycles for reordering tokens by 10.5% in simulation of 20% pruning, while providing a clearer emphasis on essential tokens (Fig.2).
- Attention Map-based Head-wise Token Sorting: Our approach differs from mainstream token selection methods [2], [3], which often overlook the varying significance of each attention head. By incorporating the variability in head importance, we fine-tune the token pruning process to effectively reduce the undesirable influence of less important heads.
- Hardware Implementation of On-the-fly Token Pruned ViT: We introduce ViT-ToGo, an optimized accelerator

979-8-3503-4859-0/DATE24/© 2024 EDAA



(a) Grouped Token Pruning (b) Individual Token Pruning

Fig. 2: Comparison between grouped and individual token pruning. Grouped token pruning provides a clearer focus with fewer steps.

designed to address the absence of real-device applications for existing token pruning methods on platforms like FPGAs. Our proposed accelerator enables parallel token sorting and subsequent computations, such as feed-forward networks. This approach effectively minimizes the time overhead typically encountered on conventional platforms, significantly enhancing the efficiency of token pruning on embedded devices.

We empirically validate our method on ViT-Base, Small, and Tiny models using the Cifar10, Cifar100, and ImageNet-1K datasets. Our experiments show that employing grouped token pruning with head-wise scoring effectively reduces computational costs while maintaining classification accuracy nearly unchanged. In the case of the ViT model, using just 30-50% of the initial tokens in the final layer results in a minimal accuracy drop of approximately 1%. Additionally, our hardware implementation results indicate that grouped token pruning requires only an average of 1.13% additional resources.

II. Preliminary

A. Vision Transformer

Vision Transformer (ViT) is an innovative architectural paradigm in computer vision, stemming from the foundational transformer model that has enjoyed widespread adoption in natural language processing (NLP). Leveraging a self-attention mechanism, ViT adeptly captures both global and local image features, showcasing competitive performance when compared to traditional Convolutional Neural Networks (CNNs), the prevailing models in the field of computer vision.

The operational sequence of ViT begins with the input image being partitioned into uniform patches, each with dimensions (P $\times$ P). These patches undergo linear transformation, resulting in vectors known as "token embeddings". Notably, a single learnable "class token" is appended at the forefront of these token embeddings, adopting a concept similar to the one used in BERT [14] for NLP. As the image data progresses through the ViT encoder towards the final output stage, this class token evolves into a comprehensive representation vector for the entire image. To incorporate spatial information, the token embeddings are further enriched with positional embeddings, which specify the position of each patch within the image.

TABLE I: The number of computations required for one encoder.

ViT Encoder Module		Number of Computations
Multi-Head Self-Attention	Query, Key, Value Embedding	$3 \times N \times D_{model} \times D_{head} \times H$
	Scaled-Dot Product Attention (Attn = Query $\times$ Key^T)	$H \times N \times D_{head} \times N$
	Attn $\times$ Value	$H \times N \times N \times D_{head}$
	Output Linear Projection	$N \times D_{head} \times H \times D_{model}$
Feed-Forward Neural Network	Fully Connected Layer 1	$N \times D_{model} \times D_{ffn}$
	Fully Connected Layer 2	$N \times D_{ffn} \times D_{model}$
Total Number of Computations per Encoder = $4NHD_{model}D_{head} + 2N^2HD_{head} + 2ND_{model}D_{ffn}$		

D_{model} : Model Dimension, D_{head} : Head Dimension, D_{ffn} : FFN Dimension
N : Number of Tokens, H : Number of Heads,

These enhanced input tokens are subsequently fed into the ViT encoder.

The encoder is composed of multiple layers, each featuring a multi-head self-attention (MHSA) mechanism followed by a feed-forward network (FFN). Prior to processing, both the MHSA and FFN inputs undergo layer normalization. The operations of the encoder are concisely described by the following equations:

$$X'_{l-1} = MHSA(LN(X_{l-1})) + X_{l-1} \quad (1)$$

$$X_l = FFN(LN(X'_{l-1})) + X'_{l-1} \quad (2)$$

Upon completion of the encoder's operations, the output is directed through a Multi-Layer Perceptron (MLP) Head, which serves the purpose of classification. Notably, only the class token is leveraged for conducting this classification task.

B. Token Pruning in ViT Accelerator

Previous approaches [2], [3], [9]–[12], [15] regarding token pruning for Vision Transformers (ViT) mainly focus on evaluating the importance of individual tokens to keep the most critical ones. However, only one study for ViT, [9], has explored its application on embedded devices. Additionally, these prior works often insert the token pruning module between layers, leading to an increase in latency that offsets the benefits of pruning.

In contrast to existing methods that focus on individual token assessment and pruning, our approach introduces a novel methodology that involves grouping adjacent tokens together and pruning based on the collective importance of these token groups. Our approach gains prominence because prior research predominantly targets general-purpose hardware like GPUs and CPUs, with limited consideration for the specific constraints of edge devices.

Furthermore, while existing methodologies typically incorporate a separate token pruning module between ViT layers, our novel hardware architecture aims to minimize latency through parallel processing. This design enhances the feasibility of deploying pruned ViT models on edge devices by maximizing latency reduction.

In the subsequent sections, we delve deeper into the proposed grouped token pruning methodology and its associated hardware implementation, showcasing its effectiveness and efficiency in making ViT models more suitable for resource-constrained edge device deployments.

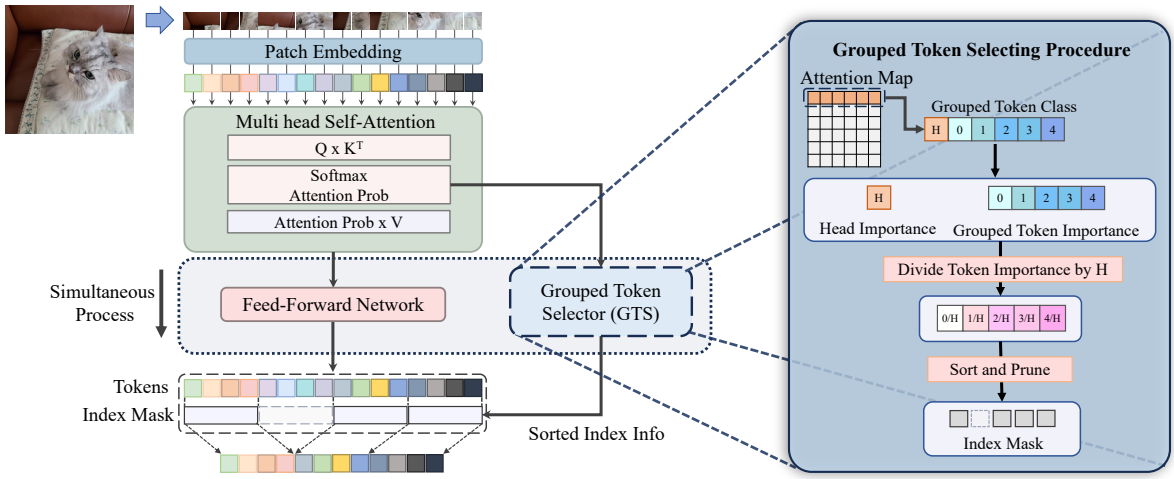


Fig. 3: Overall Procedure of ViT-ToGo. The Grouped Token Selector (GTS) utilizes attention probabilities obtained from the multi-head self-attention (MHSA). For head-wise token importance estimation, grouped token importance is divided by head importance, as smaller head importance values indicate more significant heads. It operates simultaneously with the feed-forward network (FFN) and ultimately provides preservation index information to prune unimportant grouped tokens.

III. Proposed Grouped Token Pruning with Consideration of Head Importance

A. Grouping Neighboring Tokens

Fig. 2b illustrating individual token pruning provides insights into the phenomenon where, when more than half of adjacent tokens are pruned, they often represent redundant background information. Conversely, neighboring tokens of essential tokens were mostly retained. This observation underscores the practicality of grouping and collectively pruning neighboring tokens in vision tasks.

Our Grouped Token Selector (GTS), detailed in Fig. 3, uses the first row of attention probabilities derived from the MHSA stage. This particular row holds significance as it corresponds to the class token, which gathers information from all other tokens in the sequence. We ignore the first value in this row, as it only reflects the class token's relationship with itself. The rest of the values in this row serve as importance indicators of each respective token. To form groups, we sum the attention probabilities of spatially adjacent tokens within the image, as shown in Fig. 1. This summation gives a "group importance" score for each group. Then, we rank these groups based on their calculated importance and retain only the tokens in the most significant groups. The indices of these retained tokens are then used to selectively filter the output of the FFN.

Grouped token pruning is particularly efficient in hardware operations. When conducting token pruning for each token individually, it requires sorting and reordering cycles that scale with the total number of tokens. In contrast, applying grouped token pruning reduces the number of groups to just one-fourth of the total tokens compared to the individual token pruning scenario. Consequently, sorting and reordering can be performed more rapidly. Notably, reordering is essential for the efficiency of subsequent operations, and it occurs between layers in the ViT model. Individual token pruning requires the process of identifying and removing unnecessary tokens for each token individually. However, grouped token pruning allows for simultaneous reordering of tokens within a group, significantly reducing the additional latency incurred due to token reordering.

B. Head-wise Token Importance Estimation

In token pruning using attention probabilities, token importance values are obtained from the attention probabilities of each head. However, majority of previous studies neglect to consider the varying reliability of these heads, which providing the token importance by each head's attention probabilities, when determining the final token importance. However, considering the varying reliability of each head, it is effective to incorporate head importance to achieve more accurate token pruning.

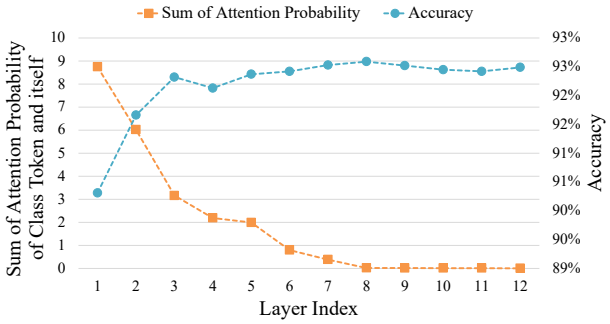


Fig. 4: Sum of attention probability of class token and itself, and accuracy of ViT-Base on Cifar100. This exhibits sum of attention probability gradually decrease with deeper layers, and accuracy decrease when GTS is inserted in shallower layer.

To address this, we introduce a head importance metric, derived from the first token in the first row of each head's attention probabilities, which is class token's attention probability. The grouped token pruning procedure is in Fig. 3.

As shown in Fig. 4, as layer goes deeper, sum of attention probabilities of class token and itself gradually decreases. When GTS is inserted in deeper layer, the accuracy improves. This shows that deeper layer is more important in recognizing important tokens. Therefore, from the fact that deeper layer have smaller class token's attention probability value, we used it as head importance score, that crucial head have smaller attention probability value. We normalize these initial head importance measures using the softmax function for better scale consistency, which is already in ViT backbone as shown in Fig 5.

This approach results in improved accuracy, substantiated by the data in Table V. Furthermore, our metric is just simply reusing previously computed value, it does not require any extra registers or time to calculate head importance score.

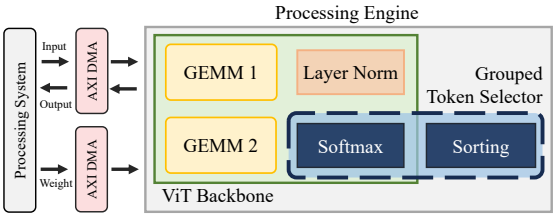


Fig. 5: ViT hardware architecture with Grouped Token Selector

IV. Hardware Implementation

A. ViT Accelerator with Grouped Token Pruning

To mitigate latency overhead arising from data movement between off-chip memory, the processing system, and the processing engine, we have employed a stream interface with DMA (Direct Memory Access) and double buffering. This approach enables direct memory access and parallelizes data transfer with computation processes. The original ViT accelerator comprises modules for general matrix multiply (GEMM), layer normalization, and softmax calculations. GELU computation is executed at the end of the GEMM operation in Fully Connected layer 1. We employ tiling for GEMM calculations to minimize the area overhead. The GTS module is incorporating the softmax module and a new sorting module.

In the GTS module, we first insert the head importance, which is attention probability values, into the softmax module to normalize head importance. Next, we add up the importance scores of grouped tokens and then divide them by their respective head importance values. This is done simultaneously for each head, resulting in the combined importance of the same tokens across all heads. (Fig. 6)

Since sorting is done in parallel with post-MHSA process, we utilized conventional merge sort algorithm. Merge sort is known as stable sort and small time complexity compared to other sorting algorithm, including quick sort, bubble sort. However, it has disadvantage in that it is not in-place sorting algorithm, which means it requires additional buffer to temporaily save data. In ViT-ToGo, we optimized merge sort by reusing of the Attention Probability Register (QK buffer). Since the QK buffer is larger than the number of groups, we freely utilized the QK

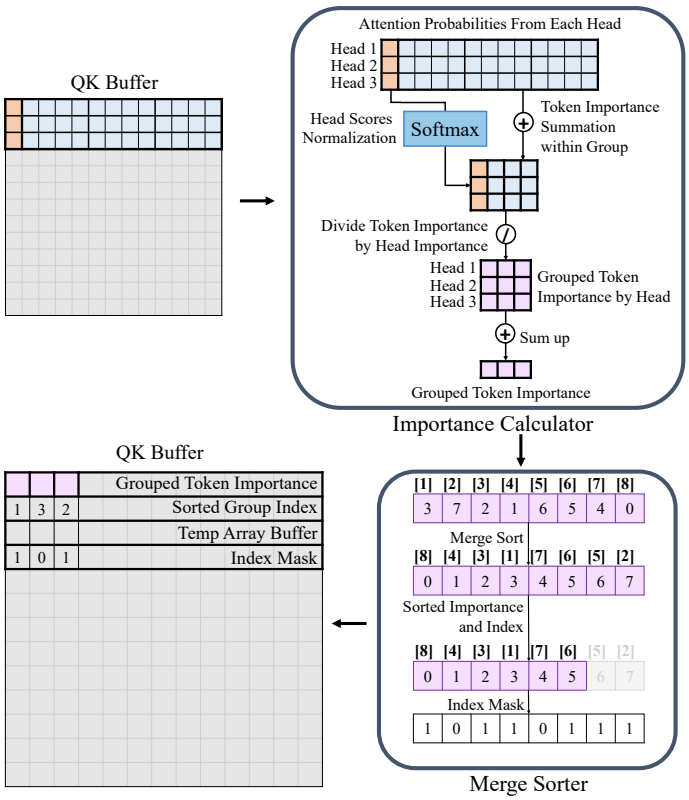


Fig. 6: Explanation of Grouped Token Sorting module. Group importance score is calculated from Importance Calculator and then sorted by Merge Sorter. Every computation is done with the QK buffer. Output index mask is also saved in the QK buffer.

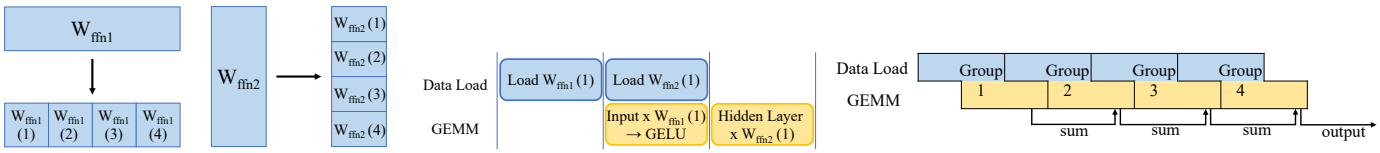
buffer for sorting, as described in Fig 6. The first row of the QK buffer first stores the group importance, while the second row serves as a temporary array. Additionally, an index mask should be created at the end of the GTS process, and it is also stored in QK buffer. By employing this approach, we proposed a ViT accelerator with GTS that has less overhead.

As mentioned in Section III-A, grouped token pruning further speeds up our proposed accelerator by accelerating the reordering process. Both the MHSA and FFN operations in ViT necessitate a skip connection process, as detailed in (1) and (2). As a result, the accelerator needs to store the input in an separate buffer. When saving to this alternate buffer, if a group is not in the index mask, the saving operation is skipped, and process moves to the next group.

B. Group-wise Feed-Forward Network Scheduling

In the case of a Feed Forward Network (FFN), the size of hidden layer weights can be represented as $D_{model} \times D_{ffn}$, or equivalently $D_{model} \times D_{model} \times 4$. However, executing FFN operations in a single step necessitates a large buffer for storing both the weights and the intermediate results. Given the challenge of data storage on resource-constrained devices, we propose a group-wise FFN scheduling method for more efficient computations on limited-resource hardware.

Similar to the operation of MHSA in a head-wise manner, the FFN weights are divided into four groups, each of dimensions $D_{model} \times D_{model}$ dimensions, as depicted in Fig.7a. Each



(a) Split the Weight of FC Layer (b) Specific flow in each group (c) Feed-Forward Network Optimization in ViT-ToGo

Fig. 7: Group-wise Feed-Forward Network Computations

group independently undergoes matrix multiplication (GEMM) operation, and their results are aggregated to form the final output. The overall flow of the FFN in ViT-ToGo is illustrated in Fig.7c. Data loading alternates between the weight group of FC layer 1 (W_{ffn1}) and the weight group of FC layer 2 (W_{ffn2}). When W_{ffn2} is loaded, matrix multiplication between input and W_{ffn1} takes place concurrently. During the loading of W_{ffn1}, the multiplication occurs between the hidden layer and W_{ffn2}. This process repeats four times, with each output being sequentially accumulated to generate the final output.

V. Experiments

A. Experimental Setup

1) Grouped Token Pruning Method: To validate our grouped token pruning method, we applied it to ViT-Base, Small, and Tiny models using Cifar10, Cifar100, and ImageNet datasets. Experiments were conducted on NVIDIA TITAN RTX and GeForce RTX 3090 GPUs. Our experimental setup aligns with the methodologies outlined in the ViT GitHub repository[1] for ViT models. We evaluated the method using three different pruning ratios: 20%, 25%, and 30%. The number of Grouped Token Selector (GTS) inserted layers ranged from 3 to 4.

2) Proposed Accelerator Hardware Platform: For the hardware implementation, we utilized Xilinx Vitis HLS 2023.1 (High-Level Synthesis), Vivado 2023.1, and targeted the ZCU104 board, which includes 624 BRAM18k blocks, 1728 DSPs, 460,800 FFs, 230,400 LUTs, and 96 LUTRAMs. We synthesized the hardware for ViT-Tiny at a 100 MHz operating frequency. The data were represented in 16-bit fixed-point format. ViT-ToGo incorporates ViT backbone modules and a sorting module.

B. Experimental Results

1) Accuracy and GFLOPs Result: The grouped token pruning method significantly reduced GFLOPs with negligible accuracy degradation across ViT-Tiny, Small, and Base models on Cifar10, Cifar100, and ImageNet datasets. Notably, the results highlight the method's effectiveness, particularly for ViT-Tiny, underscoring its suitability for smaller models. Given that edge devices often require smaller models due to limited resources, our proposed accelerator with grouped token pruning proves valuable for such resource-constrained environments.

2) Grouped Token Selector (GTS) Efficiency: To demonstrate the effectiveness of the proposed Grouped Token Selector (GTS), we integrated GTS into Evo-ViT [3]. The application of our proposed method to Evo-ViT resulted in improved throughput while maintaining accuracy levels. In Table IV, Evo-ViT with the GTS achieved enhanced throughput across DeiT-Tiny, Small, and Base models. For fair comparison of accuracy, we used pretrained weight of DeiT-Small, and finetune with each method for one epoch. Only pruning methodology is

[1]ViT baseline repository - https://github.com/jeonsworld/ViT-pytorch

TABLE II: Accuracy and GFLOPs results for ViT-Tiny, Small on Cifar10 and Cifar100 datasets. Numbers under the dataset name is the index of ViT layer where GTS is inserted.

ViT-Tiny			
	Pruning Ratio	Accuracy (%)	GFLOPs
Cifar10 (4, 7, 9, 11)	Baseline	94.91	1.26
	20%	95.18 (+0.12)	0.93 (-26.14%)
	25%	95.00 (+0.09)	0.87 (-31.04%)
	30%	94.66 (-0.25)	0.81 (-35.31%)
Cifar100 (4, 7, 9)	Baseline	80.25	1.26
	20%	81.15 (+0.9)	0.94 (-25.32%)
	25%	79.90 (-0.35)	0.88 (-30.24%)
	30%	79.33 (-0.92)	0.82 (-34.53%)

ViT-Small			
	Pruning Ratio	Accuracy (%)	GFLOPs
Cifar10 (4, 7, 9, 11)	Baseline	97.63	4.61
	20%	97.49 (-0.14)	3.43 (-25.49%)
	25%	97.18 (-0.45)	3.21 (-30.39%)
	30%	97.06 (-0.57)	3.01 (-34.72%)
Cifar100 (5, 7, 10)	Baseline	88.80	4.61
	20%	87.96 (-0.84)	3.60 (-21.70%)
	25%	87.70 (-1.10)	3.40 (-26.11%)
	30%	87.17 (-1.63)	3.22 (-30.12%)

TABLE III: Accuracy and GFLOPs results for ViT-Base on Cifar100, and ImageNet datasets. Numbers under the dataset name is the index of ViT layer where GTS is inserted.

ViT-Base			
	Pruning Ratio	Accuracy (%)	GFLOPs
Cifar100 (4, 6, 8)	Baseline	92.49	17.58
	20%	92.23 (-0.26)	12.86 (-26.83%)
	25%	91.86 (-0.63)	11.97 (-31.89%)
	30%	91.36 (-1.13)	11.14 (-36.59%)
ImageNet (4, 6, 8)	Baseline	80.05	17.58
	20%	79.60 (-0.45)	12.86 (-26.83%)
	25%	79.06 (-0.99)	11.97 (-31.89%)
	30%	78.21(-1.84)	11.14 (-36.59%)

TABLE IV: Throughput Comparison with Evo-ViT [3]. The image resolution is 224 x 224. Used the throughput benchmark from [3] GitHub repository[2].

		ViT-Tiny	ViT-Small	ViT-Base
DeiT + TG	Top-1 Acc. (%)	-	**78.80**	-
Evo-ViT + TG	Throughput (img/s)	**4773.11**	**2100.40**	**675.98**
	Top-1 Acc. (%)	-	77.10	
Evo-ViT	Throughput (img/s)	4742.74	2096.57	671.25
	Top-1 Acc. (%)	-	77.48	-

applied for comparing pruning efficiency.

Furthermore, our head-wise grouped token pruning within GTS led to increased accuracy for ViT-Tiny, Small, and Base models. As detailed in Table V, with the exception of ViT-Small with a 20% pruning ratio, accuracy improved for all other configurations. This underscores the effectiveness of head-wise importance calculation in the context of image classification tasks.

TABLE V: Head-wise Token Importance Consideration. Accuracy of ViT-Tiny, Small, Base on Cifar100. Pruning ratio per layer of first row : 20%, second row : 25%.

ViT-Tiny		**ViT-Small**		**ViT-Base**	
w/ Head	w/o Head	**w/ Head**	w/o Head	**w/ Head**	w/o Head
81.15%	80.41%	87.33%	**87.51%**	**92.22%**	91.94%
79.90%	79.85%	**86.87%**	86.29%	**91.44%**	91.38%

TABLE VI: Hardware Implementation Results of Baseline and ViT-ToGo for ViT-Tiny model, and HeatViT [9] for DeiT-Tiny model. Baseline and ViT-ToGo targeted the ZCU104, and [9] targeted the ZCU102. Since ZCU102 have different available resource, reported the absolute number for [9].

Resource (Available)	Resource Utilization (%)		
	Baseline	ViT-ToGo	HeatViT [9]
BRAM18K (624)	82.85%	83.81%	355.5 (BRAM36K)
DSP (1728)	11.40%	11.28%	1968
FF (460800)	8.01%	9.40%	126k
LUT (230400)	14.68%	18.01%	137.5k
LUTRAM (101760)	2.83%	2.96%	-
Power	3.937W	4.057W	9.453W

3) Hardware Implementation Result: The hardware synthesis results are presented in Table VI. For the ViT-Tiny model, ViT-ToGo incurs only a marginal increase of 0.96% in BRAM, 1.39% in FFs, 3.33% in LUTs, 0.13% in LUTRAMs, and decrease of 0.12% in DSPs, while significantly reducing the cycle count to 70% compared to baseline. In contrast, when compared to the DeiT-Tiny model from [9], which utilized 9% more DSPs, 4% more FFs, 8% more LUTs, and even 7% more BRAM36, ViT-ToGo demonstrates a far more efficient resource utilization. Moreover, power usage for extra grouped token pruning process makes only 1.03x compared to baseline, while [9] requires 1.18x. Additionally, it's worth noting that ViT-ToGo is designed to operate on 16-bit fixed-point values, whereas [9] employs 8-bit fixed-point values, further highlighting the efficiency of our proposed accelerator.

[2]Evo-ViT repository - https://github.com/YifanXu74/Evo-ViT/tree/main

VI. Conclusion

In this paper, we have introduced a Vision Transformer (ViT) accelerator that incorporates grouped token pruning. By adopting this approach, which groups neighboring tokens, we were able to reduce both the time required for pruning and the reordering latency in the accelerator, compared to traditional individual token pruning methods. Furthermore, our proposed method of head-wise group importance estimation enhances the overall performance of the system.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[2] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," in *Advances in Neural Information Processing Systems*, 2021.

[3] Y. Xu, Z. Zhang, M. Zhang, K. Sheng, K. Li, W. Dong, L. Zhang, C. Xu, and X. Sun, "Evo-vit: Slow-fast token evolution for dynamic vision transformer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 2964–2972, Jun. 2022.

[4] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," in *The Eleventh International Conference on Learning Representations*, 2023.

[5] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width depth pruning for vision transformers," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 3143–3151, Jun. 2022.

[6] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing sparsity in vision transformers: An end-to-end exploration," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 19974–19988, Curran Associates, Inc., 2021.

[7] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-training quantization for vision transformer," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 28092–28103, 2021.

[8] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization," in *Computer Vision – ECCV 2022*, (Cham), pp. 191–207, 2022.

[9] P. Dong, M. Sun, A. Lu, Y. Xie, K. Liu, Z. Kong, X. Meng, Z. Li, X. Lin, Z. Fang, and Y. Wang, "Heatvit: Hardware-efficient adaptive token pruning for vision transformers," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 442–455, 2023.

[10] M. Fayyaz, S. A. Koohpayegani, F. R. Jafari, S. Sengupta, H. R. V. Joze, E. Sommerlade, H. Pirsiavash, and J. Gall, "Adaptive token sampling for efficient vision transformers," in *Proceedings of the 17th European Conference on Computer Vision (ECCV)*, p. 396–414, 2022.

[11] Y. Tang, K. Han, Y. Wang, C. Xu, J. Guo, C. Xu, and D. Tao, "Patch slimming for efficient vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12165–12174, June 2022.

[12] H. Yin, A. Vahdat, J. M. Alvarez, A. Mallya, J. Kautz, and P. Molchanov, "A-vit: Adaptive tokens for efficient vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10809–10818, June 2022.

[13] Z. Song, Y. Xu, Z. He, L. Jiang, N. Jing, and X. Liang, "Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction," in *arXiv*, 2022.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pp. 4171–4186, June 2019.

[15] S. Wei, T. Ye, S. Zhang, Y. Tang, and J. Liang, "Joint token pruning and squeezing towards more aggressive compression of vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2092–2101, June 2023.