ReViT: Vision Transformer Accelerator With Reconfigurable Semantic-Aware Differential Attention

Xiaofeng Zou, Cen Chen, *Senior Member, IEEE*, Hongen Shao, Qinyu Wang, Xiaobin Zhuang, Yangfan Li, and Keqin Li, *Fellow, IEEE*

***Abstract*—While vision transformers (ViTs) have continued to achieve new milestones in computer vision, their complicated network architectures with high computation and memory costs have hindered their deployment on resource-limited edge devices. Some customized accelerators have been proposed to accelerate the execution of ViTs, achieving improved performance with reduced energy consumption. However, these approaches utilize flattened attention mechanisms and ignore the inherent hierarchical visual semantics in images. In this work, we conduct a thorough analysis of hierarchical visual semantics in real-world images, revealing opportunities and challenges of leveraging visual semantics to accelerate ViTs. We propose ReViT, a systematic algorithm and architecture co-design approach, which aims to exploit the visual semantics to accelerate ViTs. Our proposed algorithm can leverage the same semantic class with strong feature similarity to reduce computation and communication in a differential attention mechanism, and support the semantic-aware attention efficiently. A novel dedicated architecture is designed to support the proposed algorithm and translate it into performance improvements. Moreover, we propose an efficient execution dataflow to alleviate workload imbalance and maximize hardware utilization. ReViT opens new directions for accelerating ViTs by exploring the underlying visual semantics of images. ReViT gains an average of 2.3× speedup and 3.6× energy efficiency over state-of-the-art ViT accelerators.**

***Index Terms*—Hardware accelerator, vision transformers, software-hardware co-design.**

Received 20 April 2024; revised 23 October 2024; accepted 10 November 2024. Date of publication 21 November 2024; date of current version 12 February 2025. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2023ZYGXZR023, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515010220, in part by the Postdoctoral Fellowship Program of CPSF under Grant GZC20230841, in part by the China Postdoctoral Science Foundation under Grant 2024M760955, in part by the National Natural Science Foundation of China under Grant 62472181 and Grant 62302529, in part by the Basic research of Shenzhen Science and Technology Plan under Grant JCYJ20210324123802006, in part by Hunan Provincial Natural Science Foundation of China under Grant 2023JJ40770, in part by Changsha Municipal Natural Science Foundation under Grant kq2208290, sponsored by CCF-Phytium Fund, and sponsored by CAAI-MindSpore Open Fund, developed on OpenI Community. Recommended for acceptance by Y. Hu. *(Corresponding author: Cen Chen.)*

Xiaofeng Zou, Hongen Shao, Qinyu Wang, and Xiaobin Zhuang are with the School of Future Technology, South China University of Technology, Guangzhou 510641, China (e-mail: zouxiaofeng@scut.edu.cn; ftshaohongen@mail.scut.edu.cn; ft_wangqinyu@mail.scut.edu.cn; ftzxc111@mail.scut.edu.cn).

Cen Chen is with the School of Future Technology, South China University of Technology, Guangzhou 510641, China, and also with Pazhou Laboratory, Guangzhou 510335, China (e-mail: chencen@scut.edu.cn).

Yangfan Li is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: liyangfan37@csu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA.

Digital Object Identifier 10.1109/TC.2024.3504263

I. Introduction

RECENTLY, transformers [1] have emerged as the predominant method for sequence modeling tasks in natural language process (NLP) and related fields [2], [3] owing to their exceptional performance. Inspired by this, researchers began to expand the transformer architecture into computer vision, i.e., vision transformers (ViTs) [4], achieving promising performance in various fields such as image retrieval [5], semantic segmentation [4], image classification [6], [7], object detection [8], etc. To accommodate image processing, ViTs typically divide the input image into a sequence of fixed-size patches (e.g. 16×16) [9], and model the global context relationships among different patches through multi-head self-attention (MHSA). Compared to convolution neural networks (CNNs), ViTs can effectively capture long-range interactions and global information for better performance.

Despite the remarkable success of ViTs, applying them to real-world applications still poses serious challenges [10], [11]. The primary bottleneck arises from the self-attention mechanism employed by the ViT models, which requires computing the interactions among all patches, and its computational and memory complexity depends on the number of patches with a quadratic relationship [10]. This hinders its achievable efficiency and scalability, and inhibits widespread deployment on resource-constrained edge devices.

Unlike transformers in NLP, real-world images usually exhibit hierarchical semantics [12]. The human visual system instinctively leverages this hierarchical semantics during the perception process, gradually extending from local details to global context, thereby forming a comprehensive understanding of the image [13], [14]. Inspired by this, we identify three levels of visual semantics within the ViT framework. Taking Fig. 1(a)

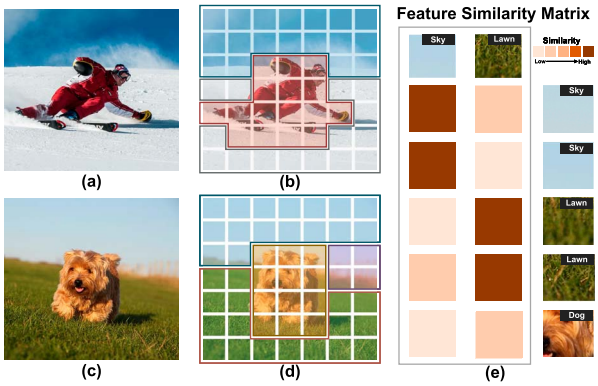

Fig. 1. The visual semantics of images. Different colors represent different semantics classes. Taking (a) as an instance, the image can be divided into multiple levels of semantic information such as sky, snow, and person. (e) Feature similarity of different semantics.

as a reference, we observe the following: (i) Hierarchical semantics: Images possess multi-level semantic layers, such as the sky, snow, and people in a skiing scene. By synthesizing this information, we gain a holistic understanding of the complex scenario. (ii) Varied scale semantics: Different semantic categories within an image manifest at various scales, leading to an uneven representation and spatial distribution. For example, the background elements like snow and sky cover a larger portion, whereas the foreground elements, such as people, are confined to smaller areas. (iii) Feature similarity within the same semantics: As shown in Fig. 1(e), objects of the same semantic class (e.g., sky and grass) tend to cluster in close spatial proximity, exhibiting strong feature similarity. Meanwhile, there are significant feature differences among different semantic classes.

To alleviate the inefficiency of ViTs, researchers have made great strides in designing customized accelerators [15], [16], [17], which significantly outperform general-purpose platforms such as CPUs and GPUs. However, these ViT accelerators remain grounded in the vanilla flattened attention mechanism. They incorporate optimization schemes such as sparse approximation or token pruning from sequence transformers in NLP, and apply them to flattened image patches. In contrast, recent studies in computer vision algorithms have demonstrated that leveraging hierarchical [4], [18] and varied-scale [19], [20], [21] visual semantics can improve accuracy by maintaining semantics in the scope of attention operation while simultaneously reducing computational workload by limiting information interactions to local windows.

Despite the promising potential of visual semantics in accelerating ViTs, it poses several significant challenges. (i) The hierarchical semantic and semantic irregularity of images would result in irregular attention computations. However, existing ViT accelerators [15], [16], [17] are limited to flattened attention computation, hindering their ability to fully leverage these semantic properties to accelerate ViTs. (ii) The varying scales of different semantic categories cause imbalances in attention workloads, and simple sequential execution leads to idle resources. (iii) Attention computations involve multiple complex operations, making it difficult to effectively exploit feature similarities within the same semantics.

In this work, we design a systematic algorithm-architecture co-design approach to end-to-end explore the hierarchical semantics of images for accelerated ViT inference. We first propose a novel semantic-aware hierarchical differential attention algorithm for ViTs. It consists of two mechanisms: semantic-ware differential attention mechanism, and efficient hierarchical differential attention mechanism. The first mechanism utilizes both semantic-aware and differential execution models to optimize attention for a specific semantic by reducing computation and communication redundancies. The second mechanism aims to support efficient hierarchical attention for different semantics. Our proposed algorithm can improve execution efficiency and maintain the same recognition accuracy as state-of-the-art ViTs, which use adaptive hierarchical attention. Moreover, we also propose a semantic-aware hierarchical pruning method to prune unimportant patches in a specific semantic and unimportant semantic.

Since the proposed algorithm involves multiple complex operations, such complex execution dataflow cannot be supported well by current general-purpose architecture and ViT accelerators [15], [17], [22]. To tackle this, we design a dedicated accelerator ReViT to support the proposed algorithm and translate it into performance improvements. Specifically, we design a *Reconfigurable Attention Engine* that explores two levels of fine-grained reconfigurability to efficiently support the semantic-aware hierarchical differential attention: (i) Processing element unit (PEU)-level reconfiguration to support multi-granularity attention computation, including regular flattened attention and irregular hierarchical attention. It can flexibly allocate computation resources according to the workload, effectively improving computation efficiency and reducing resource waste. (ii) Processing element (PE)-level reconfiguration to support the differential attention. We design a *Reconfigurable Bit-Serial Processing Element* that seamlessly accommodates both attention and bit-serial differential attention. Finally, we proposed a latency-aware out-of-order execution dataflow to alleviate workload imbalance for maximizing hardware utilization to further improve performance.

To the best of our knowledge, ReViT is the first co-design framework for exploring the underlying hierarchical visual semantics, offering a fresh perspective and opportunity to accelerate ViTs. To summarize, our contributions are as follows:

- We reveal that hierarchical visual semantics are essential properties of real-world images, and provide a comprehensive of how to leverage hierarchical visual semantics to accelerate ViTs.
- We propose a novel algorithm that can reduce the computation and communication in a differential attention mechanism, and support hierarchical attention by exploiting hierarchical visual semantics.
- We further propose a specialized reconfigurable architecture design to support the proposed algorithm and translate it into performance improvements. More importantly, ReViT is more generality than previous ViT accelerators [15], [17]. Through flexible configuration, it can

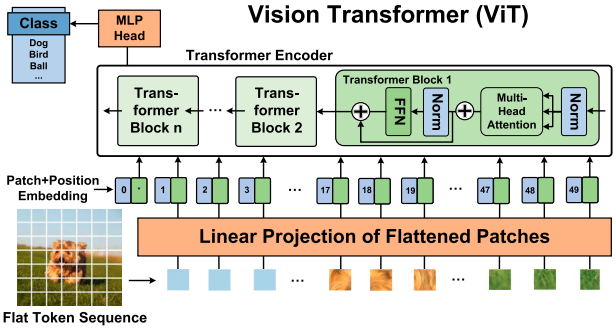

Fig. 2. Overview of ViTs with flattened attention.

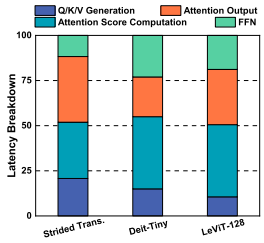

Fig. 3. Latency breakdown on A100 GPU.

effectively support different attention mechanisms including our proposed algorithm.

- Compared with ViTALiTy [17], the state-of-the-art ViT accelerator, ReViT achieves an average of 2.3× speedup and 3.6× energy-efficiency.

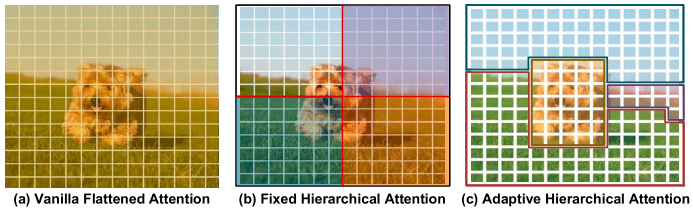

Fig. 4. Illustration of three representative ViTs leveraging the image semantics.

II. Background and Analysis

A. Preliminaries of Vision Transformer

Motivated by transformer's powerful performance, Dosovitskiy et al. [23] extended it to image classification to achieve competitive performance. Currently, ViTs have been applied to various vision domains such as semantic segmentation [4], 3D vision detection [24], image classification [6], [7], etc.

ViT Model Architecture. Fig. 2 illustrates the model architecture of ViTs. Given the input 2D image $X \in \mathbb{R}^{h\times w\times c}$, ViT flattens it into a series of image patches $X_p \in \mathbb{R}^{n\times\left(p^2\cdot c\right)}$, where c is the number of channels, (h, w) and (p, p) respectively represent the resolution sizes of the original image and patch images, and $n = hw/p^2$ is the number of patches. A linear projection module is utilized to map each image patch X_p to d-dimension to obtain the patch embedding $X_0 \in \mathbb{R}^{n\times d}$. Then, the patch embedding X_0 is fed to the L-layer Transformer block for processing. Each Transformer block includes a Multi-Head Self-Attention (MHSA) module and a feed-forward neural network (FFN). Finally, the first feature vector is sent to a feed-forward neural network to obtain the classification result.

Self-Attention in ViTs. Self-attention is the fundamental component of ViTs, which captures important global context information via modeling the dependencies between patches. It contains the following three main computation steps: (i) $Q/K/V$ **Generation**. It utilizes linear transformations (MLP) to compute the query/key/value vectors: $Q = w_Q \cdot X, K = w_K \cdot X, V = w_V \cdot X$, where w_Q, w_K, w_V are the weight parameter. (ii) **Attention Score Computation**. It computes the scaled dot-product to obtain the attention score S, and then utilizes $\mathrm{softmax}$ function to normalize it to obtain the final attention score: $A = \mathrm{softmax}\,(S) = \mathrm{softmax}\left(\frac{Q\cdot K^\top}{\sqrt{d_k}}\right)$. (iii) **Attention Output Computation**. Multiply the attention scores with the values V to obtain the final output: $Z = A \cdot V$.

B. Motivation and Analysis

1) Performance Analysis of ViTs: Despite the remarkable success of ViTs, applying them to practical applications still faces serious challenges [10], [25]. The primary bottleneck is that the self-attention mechanism adopted by ViTs requires computing the information interactions among all patches. The computation and memory complexity depend on the number of patches with quadratic relationships, especially as the input image size increases, the computational complexity would rise significantly. To better understand this phenomenon, we present the runtime distribution of various models at each stage in Fig. 3. We observe that MHSA is the key factor affecting ViT inference, which accounts for more than 70% of the total latency. Meanwhile, the attention score computation consistently dominates the MHSA runtime, accounting for 35% to 40% of the total runtime. These observations demonstrate that decreasing the computational complexity of self-attention is critical to accelerating ViTs.

2) Analysis of Semantics in ViTs: Real-world images contain the following key fundamental properties:

Hierarchical Semantics. Real-world images usually exhibit hierarchical semantics [12], and the human visual system instinctively leverages this hierarchical semantics during the perception process, gradually extending from local details to global context to form an overall understanding of the image or scene [14]. As shown in Fig. 4(a), when observing an image, humans naturally begin by focusing on its local details. Based on low-level features such as color and shape, we effortlessly identify multi-level semantic information, such as sky, lawn, dog, and combine them together to understand the entire scene.

Varied Sizes of Different Semantics. Objects of different semantic classes have different scales in the image, exhibiting irregular representation and spatial distribution. As shown in Fig. 4(c), the background (sky and lawn) occupies a large part

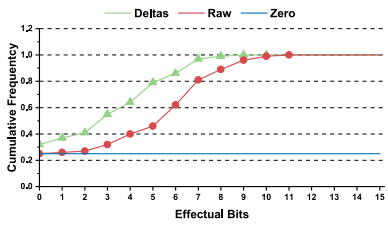

Fig. 5. Cumulative distribution of the number of effectual bits.

of the image, while the foreground (dog) takes up only a small fraction. Furthermore, the spatial distribution of these objects is arbitrary and irregular, potentially appearing in different positions and poses in the image.

Feature similarity within Same Semantics. Due to the hierarchical semantics of images, objects within the same semantics class exhibit strong feature similarity. As shown in Fig. 4(c), the features of the sky exhibit small differences in the image and demonstrate strong feature similarity within the local region. To gain insight into this property, we employ locality-sensitive hash (LSH) clustering [26] to cluster patches into several semantic groups (more details in Section III-B), and utilize the delta method [27] to quantify feature similarity. In particular, given a semantic group I_g with m patches, we compute the average of all patches to be the centroid x_c, the patch $x_i \in I_g$ can be represented in a differential form:

$$x_i = x_c + (x_i - x_c) = x_c + \Delta_i, \tag{1}$$

where Δ_i is the element-wise differences between x_i and x_c. If the original features are similar, Δ_i is small. This provides an opportunity to reduce the required computational workload. To facilitate understanding, consider a multiplication $x_i \times w$ of node x_i and weight w. If x_i is represented by p bits, the multiplication is equivalent to adding p terms, where the i-th term is the result of multiplying the s-th bit of the multiplier x_i with the shifted s bit positions multiplicand w:

$$x_i \times w = \sum_{s=0}^{s=p} x_{s_m} \cdot (w \ll s). \tag{2}$$

Only the '1' valued bits in x_i contribute to effective computation. The effectual bits correspond to the count of 1 values in x_i. Fig. 5 presents the cumulative distribution of the number of effectual bits for both original features and delta features across various neural network layers. The distribution is measured on the ImageNet-1K dataset [28] using DeiT-Base [7] with 16-bit precision. It is observed that approximately 28% of the original features have 0 effective bits, while 35% of the delta features have 0 effective bits. Meanwhile, the delta features have significantly fewer effective bits compared to the original features. This indicates that there is considerable potential to reduce the number of required computations if delta features are processed instead of raw features.

These inherent hierarchical visual semantics provide a great opportunity to improve ViTs. *It motivates us to consider the hierarchical visual semantics in self-attention, hitting two "birds" with one stone (simultaneously improving the model performance and execution efficiency of ViTs).*

3) How Existing ViTs Leverage Semantics: Earlier ViTs [6], [7] failed to consider the hierarchical semantics inherent in images. As illustrated in Fig. 4(a), these methods directly employ flattened attention, which simply splits the image into flattened patches and utilizes the self-attention mechanism to perform information interaction among all the patches. Such treating all patches equally would bring tedious redundant information interactions, thereby affecting the model's efficiency and performance.

Following this line, some approaches [4], [18] have studied fixed hierarchical attention in ViTs to explore hierarchical semantics. As shown in Fig. 4(b), these methods divide the image into fixed-size windows, then execute local attention within the windows, and then utilize global attention to realize cross-window interaction. This effectively reduces the global scope of the information interaction to local windows, thereby reducing computation redundancy and improving execution efficiency. Nevertheless, objects of different semantic classes have different scales in the image, and an object may span multiple windows. Fixed window partitions would destroy the semantic hierarchy of the image, making it difficult to capture the full local structure associated with the objects.

To explore the semantic irregularity, some works [19], [20], [21] have further proposed adaptive hierarchical attention on the basis of fixed hierarchical attention. This approach can adaptively generate partitions based on image content and capture local semantics and global context through hierarchical attention. Table IV illustrates the comparison of accuracy and throughput of different models on the ImageNet-1K dataset. It is easy to find that fixed hierarchical attention outperforms flattened attention in terms of accuracy and efficiency. Furthermore, adaptive hierarchical attention achieves higher accuracy than fixed hierarchical attention in terms of accuracy, while the efficiency is lower, but still higher than flattened attention. Therefore, it can be concluded that exploring image semantics in ViTs can effectively improve the model performance.

4) Gaps of Existing Architecture: Despite the excellent performance of adaptive hierarchical attention, it cannot be well supported by existing general-purpose architectures (e.g., GPU) and ViT accelerators [15], [16], [17]. The main challenges arise from the following aspects:

Challenge 1: Lack of exploration of feature similarity within same semantics. Feature similarity within the same semantics provides a great opportunity to improve ViT performance. However, to the best of our knowledge, neither existing algorithms nor accelerators do not exploit this feature similarity to accelerate ViTs. This may be because the attention computation in ViTs involves multiple complex operations (e.g., $Q/K/V$ generation, attention score computation, and attention output computation), making it a non-trivial task to exploit this similarity.

Challenge 2: Irregular attention computations lead to inefficient execution. Since different objects in the image have different scales, adaptive hierarchical attention would adaptively divide the image into irregular semantic groups with

TABLE I
COMPARISON TO SOTA VIT ACCELERATORS

Accelerators	Flattened Attention	Hierarchical Attention	Varied Sizes of Different Semantics	Feature similarity
Sange [22]	✓	✗	✗	✗
ViTCod [15]	✓	✗	✗	✗
ViTALiTy [17]	✓	✗	✗	✗
ReViT (our)	✓	✓	✓	✓

different sizes based on the image content, leading to irregular attention computing and workload imbalance. GPUs are inherently optimized for regular computation and cannot effectively support irregular computations [29]. Meanwhile, as shown in Table I, existing ViT accelerators [15], [16] do not consider the image semantics at all, thus they only support flattened attention computation and do not support hierarchical attention computation that considers hierarchical semantic nature and semantic irregularity.

Challenge 3: Sequential execution for irregular hierarchical attention results in idle resources. Due to the workload imbalance of adaptive hierarchical attention, naive sequential execution suffers from large waiting latency. This is because global attention must wait until the local attention of all groups has finished executing, and different semantic groups of different sizes require different execution times to execute local attention, resulting in idle computing resources.

C. Overview of Our Solutions

ReViT is a systematic algorithm and architecture co-design approach, which aims to end-to-end exploit the visual semantics of images to accelerate ViTs.

Algorithm Design. To tackle Challenge 1, we propose a novel semantics-aware hierarchical differential attention algorithm for ViTs, which adaptively generates multiple semantic groups based on the image content and captures local semantics and global context in a hierarchical attention execution. Meanwhile, it innovatively supports a semantic-aware differential attention mechanism, which leverages the strong feature similarity within the same semantic group to compute attention with a differential execution model. This method effectively reduces the computational cost and communication overhead without changing the computational result. Moreover, we further propose a semantic-aware hierarchical pruning method to prune unimportant patches in a specific semantic and unimportant semantic.

Architecture Design. On the architecture level, we design a dedicated accelerator to support the proposed algorithm and translate it into performance improvements. To tackle Challenge 2, we design a *Reconfigurable Attention Engine* that explores two levels of fine-grained reconfigurability. (i) Processing element unit (PEU)-level reconfiguration to cope with load imbalance introduced by irregular attention computation. It can flexibly allocate computation resources according to the workload, effectively improving computation efficiency and reducing resource waste. (ii) Processing element (PE)-level reconfiguration to support the semantic-aware differential attention. We design a *Reconfigurable Bit-Serial Processing Element* that seamlessly accommodates differential/normal attention computation. To tackle Challenge 3, we proposed a latency-aware out-of-order execution dataflow to alleviate workload imbalance for maximizing hardware utilization to further improve performance.

III. ALGORITHM DESIGN OF REVIT

A. Overview of Existing Adaptive Hierarchical Attention

Existing adaptive hierarchical attention [19], [20], [21] contains three main phases. It first utilizes the locality-sensitive hashing (LSH) algorithm [30] to adaptively partition the image into multiple semantic groups with similar features. Then, hierarchical attention is performed for these semantic groups. within each semantic group, intra-group local attention is performed to capture local features. To enable the model to capture global dependencies, inter-group global attention is further introduced to enable information interaction between semantic groups.

B. Semantic-Aware Hierarchical Differential Attention

Despite the adaptive hierarchical attention achieves state-of-the-art recognition accuracy, its execution efficiency still has a large room for improvement because of the challenges which are discussed in Section II-B4. To this end, we propose a novel semantic-aware hierarchical differential attention, which can improve execution efficiency and maintain the same recognition accuracy as adaptive hierarchical attention. It mainly contains semantic-aware differential attention and an efficient hierarchical differential attention mechanism.

1) Semantic-Aware Differential Attention Mechanism: As described above in Challenge 1, it is significantly challenging to utilize feature similarity in ViTs. There are two reasons: (i) Objects in images exhibit irregular distribution, making it challenging to convert similar features into deltas. This requires efficiently dividing semantic groups of images and determining their centroid. (ii) Attention computation involves several complex operations, including $Q/K/V$ generation, attention score computation, and output computation. To overcome these issues, we design a novel semantic-aware differential attention mechanism. Specifically, we first partition images into several semantic groups, and compute the centroid for each semantic group to generate deltas. Then, end-to-end differential attention is carefully designed to significantly reduce the workload of the attention computation without changing the computational result.

Semantic-aware Adaptive Partition. Consistent with [21], we employ the efficient and hardware-friendly LSH to partition images into semantic groups. Its core idea is that similar data points have a high probability of falling into the same bucket in the hash space. In particular, given an input patch matrix $X \in \mathbb{R}^{n \times d}$, we utilize the hash function to map each patch $x_i \in X$ to the hash space, and generate the unique hash code:

$$H(x_i) = \lfloor (\alpha \cdot x_i + \beta)/\gamma \rfloor, \tag{3}$$

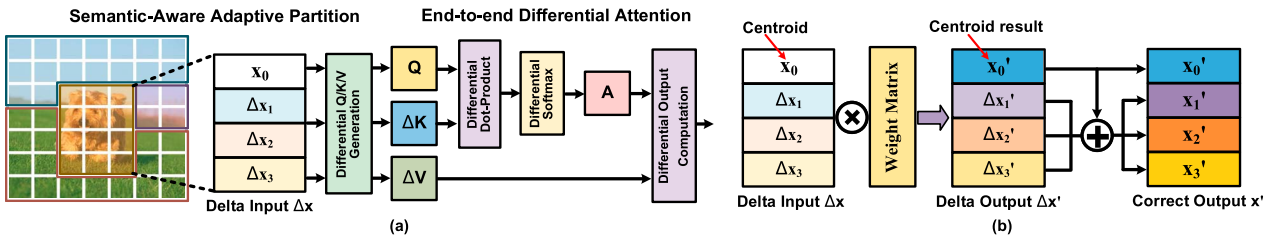

Fig. 6. (a) Illustration of semantic-aware differential attention mechanism. (b) Illustration of differential computation with delta.

where α is a vector randomly selected from the normal distribution $\mathcal{N}(0,1)$, β is randomly selected from uniform distribution $\mathcal{U}(0,\gamma)$, and γ is the width of the hash buckets. $H(x_i) \in \mathbb{R}^G$ is the hash code of x_i. This process ensures that patches with similar features have a higher probability of sharing the same hash code. For simplicity, we transform the hash codes into semantic group category indexes by:

$$idx_{x_i} = argmax\left(H(x_i)\right), \tag{4}$$

where idx_{x_i} is the cluster index for patch x_i. In this manner, the image is effectively partitioned into G semantic groups, i.e., $X = \{I_g\}_{g=1}^{G}$. Then, we take the average of all patches $x_g \in I_g$ in the semantic group I_g as the centroid x_{c_g}, and convert the patch features into deltas with Equ. 1:

$$x_{c_g} = \frac{1}{l_g}\sum\nolimits_{i \in I_g} x_{i_g}, \quad \Delta x_{i_g} = x_{i_g} - x_{c_g}, \tag{5}$$

where l_g is the number of patches in I_g.

End-to-end Differential Attention. Since the attention mechanism involves several complex operations, including linear transformations and nonlinear activations, it is not trivial to perform the attention computation in differential form. To this end, we design end-to-end differential attention to explore the computation reduction in all computation steps of attention. Fig. 6(b) illustrates the overall execution flow of end-to-end differential attention, which contains four main phases:

Step1: Differential $Q/K/V$ Generation. As shown in Fig. 6(a), given a semantic group I_g, we first utilize raw features to compute for the centroid, while the remaining patches $x_{i_g} \in I_g$ are computed in a differential manner. Take Q as an example, the process can be expressed as follows:

$$Q_{x_{c_g}} = w_Q \cdot x_{c_g}, \quad \Delta Q_{x_{i_g}} = w_Q \cdot \Delta x_{i_g}. \tag{6}$$

Finally, we add the result of centroid x_{c_g} to the other incremental results to reconstruct the correct result, i.e., $Q_{x_{i_g}} = Q_{x_{c_g}} + \Delta Q_{x_{i_g}}$. The computation of K/V follows a similar computation process. The important difference is that the K/V is stored as deltas (i.e., $\Delta K_{x_{i_g}}$ and $\Delta V_{x_{i_g}}$) without being reconstructed as correct results. The benefit of this comes from two aspects: (i) In the subsequent attention computation flow, the deltas can be reused directly without recomputation. (ii) Utilizing deltas reduces on-chip storage and communication.

Step2: Differential Attention Dot-Product. Due to the limited memory and computing resources, it is necessary to calculate the attention dot-product in blocks. Therefore, we load the features of Q/K in blocks, where Q is represented by the raw features and K is represented by the deltas. The attention score $S_{x_{i_g}}$ is then computed in an incremental manner analogous to Equ. 6, and reconstruct the correct result:

$$S_{x_{c_g}} = Q_{x_{c_g}} \cdot K_{x_{c_g}}, S_{x_{i_g}} = Q_{x_{i_g}} \cdot \Delta K_{x_{i_g}} + S_{x_{c_g}}. \tag{7}$$

Step3: Differential Softmax. Next, we perform $\mathrm{softmax}$ operation to normalize the attention score:

$$m\left(s\right) = \max_{s_i \in S}\left(s_i\right), A = \mathrm{softmax}\left(S\right) = \frac{e^{s_i - m(s)}}{\sum_{s_i \in S} e^{s_i - m(s)}}. \tag{8}$$

Nonetheless, the block computation of $\mathrm{softmax}$ presents a non-trivial challenge. This is because the normalization factor (denominator) of $\mathrm{softmax}$ contains the summation term associated with all the elements, i.e., $l\left(S\right) = \sum_{s_i \in S} e^{s_i - m(s)}$, which requires the complete attention score vector S. Therefore, the normalization operation must wait until all blocks have been executed.

To address this issue, we propose a novel differential softmax to support the blockwise computation of $\mathrm{softmax}$. Its core idea is to reuse the intermediate results of each block to obtain the final attention score in an iterative update manner. ❶ Given the attention score vector S_1 of a block, we compute the local intermediate results according to Equ. 8, and store the maximum value $m\left(s_1\right)$ and summation term $l(S_1)$. ❷ When processing the next block S_2, we update the current global maximum $m_g = \max\left(m\left(s_1\right), m\left(s_2\right)\right)$ and the global summation term $l_g = e^{m(s_1) - m_g} l\left(S_1\right) + e^{m(s_2) - m_g} l\left(S_2\right)$, and calculate the intermediate result for the current block. Then, the previous results of the old block are updated based on the new global maximum and global summation term. ❸ Until all blocks are processed, the softmax values of all blocks at this point are true values. In this manner, blockwise computation of softmax can be efficiently implemented by storing only two additional statistical values (m_g and l_g), which effectively reduces waiting latency and enhances execution efficiency.

Step4: Differential Output Computation. Finally, we load the features of A/V, where A is the raw features and V is the deltas. Then, perform $A \cdot V$ computation in incremental form and reconstruct the correct result to obtain the final output:

$$Z_{x_{c_g}} = A_{x_{c_g}} \cdot V_{x_{c_g}}, \quad Z_{x_{c_g}} = A_{x_{i_g}} \cdot \Delta V_{x_{i_g}} + Z_{x_{c_g}}. \tag{9}$$

Overall, end-to-end differential attention requires only once calculation of deltas and utilizes delta for computation, storage, and communication, thus minimizing computation, storage, and communication overheads.

2) Efficient Hierarchical Differential Attention Mechanism: Here, we integrate the proposed differential attention into adaptive hierarchical attention [21], which consists of two main phases. (i) Intra-group differential attention. Due to the strong feature similarity within semantic groups, we adopt the proposed differential attention to perform intra-group attention for capturing local features. (ii) Inter-group normal attention. Because of the large feature differences among semantic groups, we use the feature of each semantic group's centroid to perform normal attention to capture global dependency. This approach can greatly reduce the redundant information interaction between different semantic groups.

Latency-aware Out-of-Order Execution for Hierarchical Attention Process. For the issue of large waiting latency (Challenge 3) as mentioned in Section II-B4, we propose a latency-aware out-of-order execution scheme to reduce waiting latency during the hierarchical attention process. The primary insight is the insensitivity property of the ViT model to token sequences. Specifically, modifying the execution order of patches in attention would not affect the model accuracy. The latency-aware out-of-order execution scheme allows semantic groups that have finished executing intra-group attention to advance to inter-group computation. This fully pipelined execution flow minimizes the global attention waiting time. More detailed execution flow can be found in Section IV-F.

C. Semantic-Aware Hierarchical Pruning

Existing patch pruning methods [11], [31], [32] primarily focus on removing uninformative patches to reduce the computational cost of ViTs. Building upon this, we present a semantic-aware hierarchical pruning to further improve efficiency. Unlike previous methods, our approach leverages the hierarchical visual semantics to adaptively distinguish foreground and background, and prune redundant information in the foreground region. It comprises two main phases: (i) Global group pruning. To reduce extra parameters introduced by the model, we adopt the method in [32], which computes the importance of each semantic group based on class token attention. Specifically, during inter-group attention, an importance score vector is obtained by computing the interaction between class token and the centroid of each semantic group. The foreground regions with high scores are then retained, limiting intra-group attention computations to these retained regions. (ii) Local patch pruning. Similarly, we utilize class token attention to determine the importance of each patch within the semantic group, retaining the informative patches. To minimize information loss, we employ the patch packing technique [11] that summarizes non-informative patches into a package patch instead of completely discarding them.

D. Computational Complexity Analysis

For clarity, we discuss the computational complexity of the proposed ReViT here. The main additional computational overhead introduced by ReViT is semantic-aware adaptive partition. It can be sequentially decomposed into matrix multiplication, matrix addition, and the operation of finding the index of the maximum value (Eq. 3 and Eq. 4), with complexities of $O(nGd)$, $O(nGd)$, $O(nG)$ respectively. The total computational complexity is $O(2nGd + nG)$. Since the hash length G is small, the partition overhead is trivial relative to the quadratic complexity of the attention.

The reduced computational overhead comes from multiple sources: (i) During the attention score computation step, the complexity of the multiply-accumulate (MAC) operation is reduced from $O(n^2d)$ to $O(G^2 + d\sum_{i=1}^{G} m_i^2) = O(d\sum_{i=1}^{G} m_i^2)$. Notably, $\sum_{i=1}^{G} m_i = n$, therefore $\sum_{i=1}^{G} m_i^2 < n^2$, $O(d\sum_{i=1}^{G} m_i^2) < O(n^2d)$. (ii) Through hierarchical pruning, the number of patches is reduced from n to $k(k << n)$. As k decreases, both intra-group and inter-group attention computations are significantly reduced. Furthermore, as the number of ReViT layers increases, the pruning rate $\eta\%$ also rises, leading to even more pronounced computational savings.

IV. Architecture Design of ReViT

A. The Need for a Customized Accelerator

Despite the proposed algorithm can accelerate ViTs, it may not be well supported by existing general-purpose architectures (e.g., GPUs) and ViT accelerators [15], [16], [17], making it difficult to translate the theoretical savings into actual performance gains. The reasons arise from the following aspects: (i) Neither GPUs nor existing ViT accelerators support the proposed semantic-aware hierarchical attention well. GPUs are inherently optimized for regular computation and cannot effectively support irregular scale attention computations. Meanwhile, existing ViT accelerators [15], [16] are tailored for flattened attention, and do not consider the hierarchical visual semantics in images at all. (ii) The semantic-aware differential attention employs the bit-serial incremental execution model, which requires processing data bit-by-bit. Neither GPUs nor existing ViT architectures can support this operation well. Furthermore, end-to-end differential attention contains multiple complex operations, and it is a non-trivial task to efficiently support such complex execution dataflow. Therefore, it is necessary to design a customized architecture accelerator.

B. Architecture Overview

Design Overview. Fig. 7 illustrates the overall architecture of ReViT, which integrates three dedicated computing engines: *Reconfigurable Attention Engine*, *Semantic Group Generation Engine*, *Pruner*. Different computing engines are designed for the different computing components, and specific hardware optimizations are applied according to their computing properties. ReViT follows a semantic-aware block execution model, which divides the image into multiple semantic groups and performs the attention computation in parallel. The patches within the same semantic group show strong similarities, which allows the designed engine to exploit these similarities to reduce computation and communication redundancy.

The *Reconfigurable Attention Engine* is the core computing component of ReViT, which explores two levels of fine-grained reconfigurability to efficiently support the complex

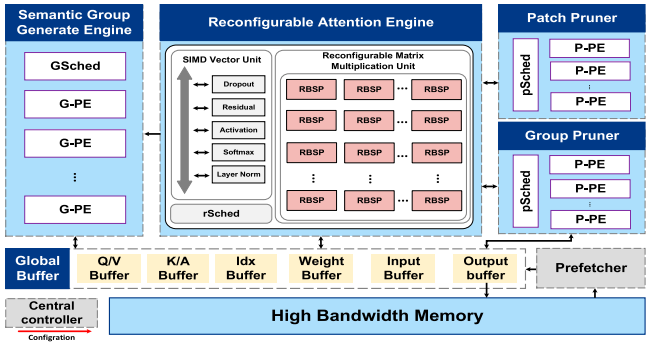

Fig. 7. Architecture overview of ReViT.

execution dataflow of semantic-aware hierarchical differential attention: (i) Processing element unit (PEU)-level reconfiguration to tackle the challenges of irregular attention computation. The designed differential attention engine can flexibly configure and allocate computational resources according to the workload. For large semantic groups, we can allocate more computation resources, while reducing resources for smaller semantic groups. This unique PEU-level reconfigurability provides a high degree of freedom to support multi-granularity attention computation (both regular flat attention and irregular hierarchical attention). Meanwhile, it maximizes the utilization of available resources to improve execution efficiency and reduce resource waste. (ii) Processing element (PE)-level reconfiguration for supporting bit-serial differential computation. We design a reconfigurable bit-serial processing element (RBSP) that can change the execution mode of PE through fine-grained configuration. This design facilitates RBSP to seamlessly accommodate both normal attention and bit-serial differential attention.

The *Semantic Group Generation Engine* is a lightweight module that aims to adaptively generate semantic group indexes for patches. It receives hash codes generated by *Reconfigurable Attention Engine* and generates cluster indexes using group-generation processing elements (G-PEs).

The *Pruners* aims to support the proposed semantic-aware hierarchical pruning. We equip ReViT with two pruning engines: *Patch Pruner* for supporting local patch pruning and *Group Pruner* for global group pruning. These two *Pruners* utilize the same pruning elements (P-PEs) for pruning.

Central Controller. ReViT is a flexible and reconfigurable pipeline architecture accelerator. The *Central Controller* can control all configurable units, which are configured in real-time based on configuration parameters to match the target execution flow.

Buffer Management. The *Prefetcher* is first utilized to explicitly prefetch the patches and weights from the high bandwidth memory (HBM). We utilize *Global Buffer* to cache various data and intermediate results to reduce the data transfer latency. In particular, *Weight Buffer* and *Input Buffer* are used to store the weight parameters and input patch features. *Idx Buffer* is used to store the group indexes of patches. *Q/V Buffer* is used to store the Q/V vectors and *K/A Buffer* to store the K-vectors and attention scores A. The *Global Buffer* adopts double-buffering technology to overlap the data transfer time with computation to hide the access latency. Note that the *Global Buffer* is a multi-mode buffer that consists of multiple scratch banks. In this manner, it allows adaptive partitioning into multiple independent buffers to match the target dataflows, and cooperate with *Reconfigurable Attention Engine* for efficient computation.

C. Design of Reconfigurable Attention Engine

The proposed hierarchical attention mechanism would divide images into irregular semantic groups with different sizes, resulting in irregular attention computation. The computing unit of existing ViT accelerators [15], [16] only supports flat attention computation. This fixed-scale computing mode does not fit well when the scale of the computation changes. It would suffer from a mismatch of computing resources, which reduces the computing resource utilization and execution efficiency. To this end, we design a flexible *Reconfigurable Attention Engine*.

Architecture of *Reconfigurable Attention Engine*. As shown in Fig. 7, it contains a scheduler *rSched* for workload assignment, a *Reconfigurable Matrix Multiplication Unit* (RMMU) responsible for supporting all matrix multiplications in attention computation, and an *SIMD Vector Unit* is used to support softmax, layernorm, dropout, and other operations.

To achieve PEU-level reconfiguration, our novel insight is to implement an omnidirectional connectivity pattern of processing elements to support shape-flexible matrix multiplication. The opportunity exists to support this pattern by adding low-cost switch boxes (MUXs) to each processing element (RBPE), as shown in Fig. 8(a). Specifically, these switch boxes are specifically used to control the data flow and determine whether the PE can send intermediate results to its right. The same single bit (1: true, 0: false) controls these switch boxes, it connects neighboring PEs if the control flow is 1, and breaks the connection if the control flow is 0. Since the feature dimensions are consistent for all patches, only configuring the columns of the computation engine suffices. In this manner, we can divide the *Reconfigurable Matrix Multiplication Unit* into multiple flexible-shaped independent sub-RMMUs through fine-grained reconfiguration.

Since attention computation involves softmax operations, if all sub-RMMUs share a single *SIMD Vector Unit*, it would cause the resource conflict problem. To support efficient parallel computation among sub-RMMUs, *SIMD Vector Unit* also needs to be decomposed into smaller components and coupled with each sub-RMMU. Due to the parallelism of the unit, we divide the original *SIMD Vector Unit* into sub-blocks proportional to the number of sub-RMMUs, and allocate each sub-RMMU with a small *SIMD Vector Unit*.

Flexible reconfigurability for Supporting Various Attention Variants. Fig. 8(a) presents an example that divides *Reconfigurable Attention Engine* into three differently sized sub-RMMUs for supporting irregular adaptive hierarchical attention. Besides, we can divide *Reconfigurable Attention Engine* into multiple fixed-sized sub-RMMUs for supporting fixed hierarchical attention. Furthermore, we configure all switch

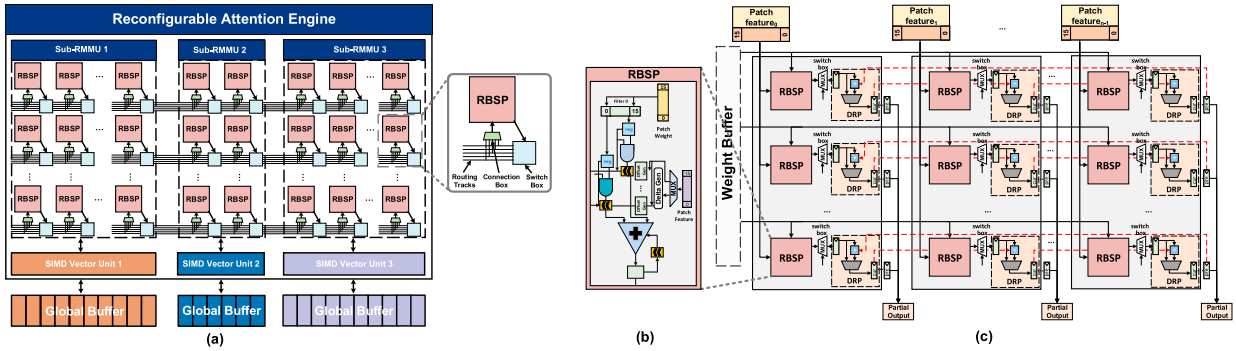

Fig. 8. (a) Configuration example of *reconfigurable attention engine*. (b) Internal of *reconfigurable bit-serial processing element*. (c) Architecture design of *reconfigurable matrix multiplication unit*.

boxes to 1, and the whole *Reconfigurable Attention Engine* as a single large execution engine for supporting flat attention Therefore, this fine-grained reconfigurable execution mode can provide tailored computational resources based on task requirements, effectively improving hardware resource utilization and execution efficiency.

D. Reconfigurable Bit-Serial Processing Element

Semantic-aware differential attention employs the bit-serial incremental execution model, which requires processing data bit-by-bit. Neither GPUs nor existing ViT architectures [15], [16] can support this operation well. Furthermore, the proposed hierarchical differential attention needs to support both differential attention and normal attention.

Architecture of *Reconfigurable Bit-Serial Processing Element*. To achieve PE-level reconfiguration, we design a novel *Reconfigurable Bit-Serial Processing Element* (RBSP), which flexibly switches between differential and normal attention computation. Specifically, it is based on the classical Bit-Pragmatic accelerator (PRA) [33], which processes the input features in a bit-serial manner, focusing solely on the effectual bits. Therefore, the execution time of the PRA is proportional to the number of effectual bits of the features. According to the analysis in section II-B, the effective bit of delta is much smaller than that of the original feature due to the local feature similarity of the image. We enhance the PRA's basic architecture to accommodate delta processing, thus leveraging this phenomenon for performance improvements.

Fig. 8(b) illustrates the internals of the RBSP. Each RBSP contains a MUX, a delta generator, offset generators, a P_c input adder tree, and P_c shifters (not multipliers). We utilize a MUX to control the dataflow of RBSP to support both differential and normal attention. It allows RBSP to compute using delta or raw feature (0: raw value, 1: delta). If MUX inputs 1, the delta generator generates the delta by subtracting element by element. The offset generator converts the delta/raw feature streams with an effective power of 2 through the modified booth encoding. In each cycle, each offset controls a shifter, effectively multiplying the weight by a power of 2. The shifted weights are reduced by the adder tree. Fig. 8(c) details the micro-architecture of sub-RMMU that contains $P_n \times P_c$ basic RBSP. Each RBSP column deals with a vector, and RBSP(i,j) deals with j-dimension features of the i-th patch. In each cycle, we read the P_c dimensional features of P_n patches to be processed. The patch feature and weight are broadcast to the relevant RBSP for matrix multiplication. To support processing deltas, each RBSP is configured with a differential reconstruction processing element (DRP) for reconstructing the final output post-delta processing. The MUX enables RBSPs to compute output features using either deltas or raw feature values. The *output reg* caches the output features.

Configurable Pipelined Execution Flow for Supporting End-to-end Differential Attention. Fig. 9 illustrates the specific execution procedure of end-to-end differential attention. Given a semantic group I_g waiting to be executed, ❶ RMMU first prefetches patch feature $X \in I_g$ based on the group index cached in *Idx Buffer* and convert it into deltas ΔX. ❷ Configure RMMU to perform differential $Q/K/V$ generation. The result is then written to the corresponding on-chip buffer, where the features of Q are stored in the correct results, while K/V are stored in deltas. ❸ Load the features of Q/K from the on-chip buffer by blocks, and dot-product computation of $Q \cdot \Delta K$ is performed in increment incremental form to obtain the attention score vector S of the block, ❹ The attention score vector is fed into *SIMD Vector Unit* to execute the differential $\mathrm{softmax}$ to obtain the intermediate results. ❺ Traverse the Q/K by blocks and repeat ❸-❹ until all the Q/K have been processed. At this time, we can obtain the normalized attention score A and write it to *A Buffer*. ❻ Load $A/\Delta V$ from the on-chip buffer, and perform the computation of $A \cdot \Delta V$ in differential form to obtain the final output.

In this manner, we utilize deltas for processing, storing, and communicating, and achieve the full-step optimization of differential attention. This greatly reduces the computational cost, storage cost, and communication overhead without compromising the accuracy of attention computation.

E. Other Engines

1) Semantic Group Generation Engine Design: Previous ViT accelerators [15], [17] do not consider the hierarchical semantics in images and thus cannot support semantic-aware adaptive partition. To achieve this, we devise the novel *Semantic*

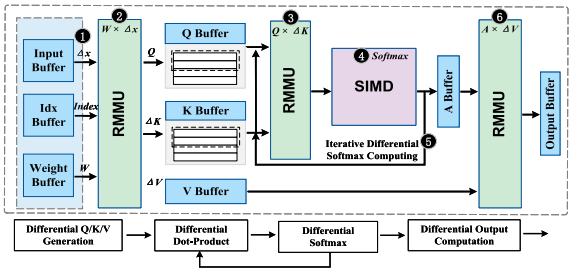

Fig. 9. Execution flow of end-to-end differential attention.

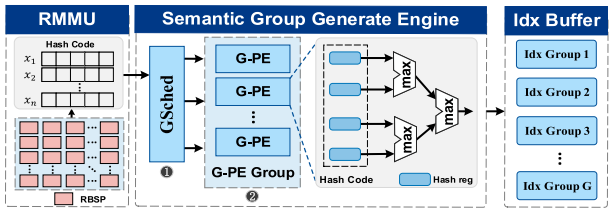

Fig. 10. Architecture design of *semantic group generation engine*.

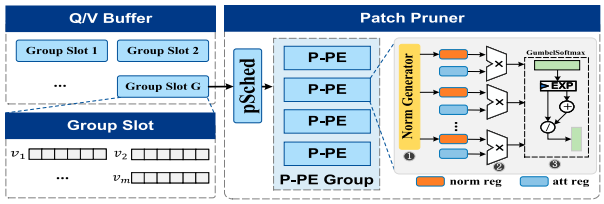

Fig. 11. Architecture design of *patch pruner*.

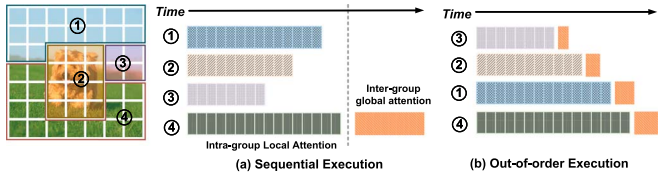

Fig. 12. Illustration of *latency-aware out-of-order execution flow*.

Group Generation Engine to support adaptive clustering. It consists of two core components: a task scheduler (gSched) and a set of group-generation elements (G-PEs). Fig. 10 illustrates the workflow of *Group Partition Engine*. It receives the patch hash codes generated by the *RReconfigurable Matrix Multiplication Unit* and assigns the workloads to each G-PE by gSched. G-PE is designed to convert hash codes into semantic group indexes. A single G-PE is implemented by multiple ALUs and registers. Given the hash code for a patch, the execution flow of a single G-PE is as follows: Each *hash reg* in G-PE reads one-dimensional hash code and then performs max operations in parallel with multiple ALUs to obtain the final semantic group index, and store it in *Idx Buffer*.

2) Pruner Design: The *Pruner* aims to support adaptive hierarchical pruning. Since local patch pruning and global group pruning have the same operation, two *Pruners* share the same micro-architecture. Fig. 11 presents the architecture of *Pruner*, which includes a task scheduler (psched) responsible for assigning tasks, a group of pruning processing elements (P-PEs) for performing the pruning operations and returning important/unimportant patch mask. Each P-PE contains a norm generator, several registers, ALUs, and a GumbleSoftmax unit. Given a set of patches to be pruned The execution process of the single P-PE is illustrated as follows: ① When pruning starts, the norm generator obtains the V vectors of the corresponding patches from the *Q/V Buffer*, performs the norm computation, and stores the result into the *norm reg*. Meanwhile, each *att reg* caches attention maps produced by the RMMU. ② Multiple aliases are utilized to perform parallel multiplication operations to obtain the importance score of each patch. ③ Apply the GumbelSoftmax unit to convert it to token retention/pruning decision masks.

F. Latency-Aware Out-of-Order Execution Flow

Due to the imbalanced workload of semantics-aware hierarchical attention, a simple sequential execution would introduce a new problem: Different semantic groups of different sizes require different execution times to perform the local attention computation, smaller semantic groups end the computation earlier, and larger semantic groups require more execution time. Unfortunately, global attention must wait for all groups' local attention to complete before it can be executed. In this case, simple sequential execution leads to large waiting latency, thereby resulting in idle computational resources.

Fig. 12(a) illustrates an example of sequential execution. The complex semantic group ④ contains more patches than other semantic groups (①,②,③), and its intra-group local attention computations require longer processing times. At this time, the computational resources used for other semantic groups (①,②,③) remain idle until ④ is executed.

To tackle this issue, we exploit the insensitivity property of the ViT model to token sequences and propose a latency-aware out-of-order execution scheme, as shown in Fig. 12(b). Attention computation is utilized to capture the interactions between different patches in the input sequence, and modifying the order of attention computation will not affect the model accuracy. Based on this property, we can reorder the attention computation for each semantic group based on the processing latency of the semantic groups. Due to the small size of the semantic group ③, it would early finish the computation of intra-group attention. Once the computation of semantic groups ③ is completed, we can reconfigure the spare hardware resources for inter-group attention computation. From then on, every time a new semantic group finishes the computation, the inter-group attention computation can be performed until all semantic groups (②,①,④) have been processed.

Through the latency-aware out-of-order execution, global attention can be computed in advance without waiting for all local attention to be computed. This fully pipelined execution flow can minimize the waiting time for inter-group global attention,

TABLE II
AREA BREAKDOWN OF REVIT

Module	Area (%)
Reconfigurable Attention Engine	61.4
Pruner	10.2
Semantic Group Generation Engine	4.8
Total Buffer	21.0
Others	2.6

and effectively improve hardware utilization to enhance the overall execution efficiency.

V. EVALUATION RESULTS

A. Experimental Setup

Evaluated Models and Dataset. Our evaluations cover various prominent ViT models, including standard ViTs, e.g., DieT-Base/Small/Tiny [7], ViT variants for mobile devices, e.g., LeViT-128s/128 [6], and ViT variants with fixed hierarchical attention, e.g., Siwn-Small [4], FasterViT [18] achieving SOTA performance using hierarchical attention. Various methods are benchmarked with: (i) vanilla flattened attention [7], (ii) fixed hierarchical attention [4], (iii) sparse attention [22] (with a sparsity threshold of 0.02), (iv) linear attention [17], (v) ReViT-w/o pruning and ReViT. Consistent with existing ViT accelerators [16], [17], we conduct experiments on ImageNet-1K dataset [34] in image classification tasks, and COCO dataset [35] in object detection task.

Compared Baselines. We compare the performance of our ReViT accelerator with two kinds of hardware baselines. The first category is the general platforms, including the CPU platform featuring Intel Xeon(R) CPU E5-2680 v3 CPUs with 500GB DRAM, and the GPU platform with NVIDIA A100. The second category consists of the SOTA ViT accelerators, including Sanger [22], ViTALiTy [17], and ViTCoD [15]. For a fair comparison, we follow [15] to enhance the hardware resources of ReViT, so that their hardware budgets are comparable to the above baseline. To facilitate performance comparisons with the baseline accelerator, we extend our accelerator to support the dataflow of the baseline accelerator, i.e., Sparse (Sanger and ViTCoD) and Linear (ViTALiTy).

Software Implementation. We implement our proposed semantic-aware hierarchical attention mechanism on each baseline ViT model using Pytorch [36], and follow [37] to employ quantization-aware training, which reduces the precision of the input features to 8 bits without sacrificing accuracy. We finetune the model using the same training strategy as in [7] and [18], training 300 times using the AdamW optimizer [38], with a learning rate of 5e-4 and a total batch size of 1024. To facilitate comparison with fixed-window hierarchical attention, we set the number of semantic groups to 4. We profile the energy consumption of CPU with PyRAPL, and the running power of GPU is estimated using PyNVML.

Hardware Implementation. To evaluate the performance of ReViT, we develop a validated custom cycle-accurate simulator to get the cycle numbers. The simulator models the micro-architectural behavior of each module, integrating with Ramulator [39] for HBM timings estimation and command trace generation. For each module's power and area measurements, we implement them in Verilog and employ Synopsys Design Compiler (DC) with TSMC's 40 nm technology for synthesis, estimating power consumption with Synopsys PrimeTime PX. Additionally, we employ CACTI [40] to compute the on-chip memory's area and power. The total area of ReViT is 15.1 mm^2. Table II shows the computing components' area percentages. Detailed configurations are provided in Table III.

TABLE III
SYSTEM CONFIGURATIONS OF COMPARED ACCELERATORS

Chip	Sanger	ViTALiTy	ViTCoD	ReViT (Ours)
Cores	64 × 64 Systolic MAC Unit Array	64 × 64 Systolic MAC Unit Array	128 MAC lines with each having 32 MACs	64 × 64 RBSP Array
SRAM(KB)	512	200	576	660
Area (mm2)	13.4	14.5	13.9	15.1
Frequency	1 GHz	1 GHz	1 GHz	1 GHz
DRAMBandwidth	HBM 2256 GB/s	HBM 2256 GB/s	HBM 2256 GB/s	HBM 2256 GB/s
Technology	40 nm	40 nm	40 nm	40 nm

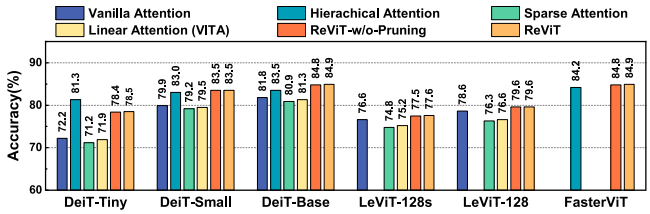

Fig. 13. Accuracy comparison among ReViT and baselines on typical models.

B. Algorithm Performance

Accuracy. Fig. 13 demonstrates the accuracy comparison between ReViT and the baseline on typical models. It is easy to observe that: (i) The proposed ReViT outperforms the baseline in almost all the settings. This can be attributed to the fact that ReViT fully leverages the hierarchical semantic of images to reduce redundant information interactions for better accuracy. Furthermore, it is also effectively illustrated that the proposed method is a plug-and-play module that can be integrated into different ViTs to achieve better performance. (ii) Compared to fixed hierarchical attention, both our ReViT-w/o-pruning and ReViT achieve better performance. In particular, ReViT improves a 0.7% performance gain compared to FasterViT. The main reason is that the proposed semantic-aware hierarchical attention can better capture the complete semantic information of objects within images and adapt to the irregularity of object scales for more robust recognition. (iii) Our ReViT demonstrates considerable improvement, ranging from 2.8% to 7.3% on sparse settings, and slightly improved accuracy compared to ReViT-w/o-pruning. This is because the semantic-aware hierarchical pruning can focus on critical foreground patches and effectively eliminate irrelevant details, thus improving the model's efficiency while preserving important information.

TABLE IV
ACCURACY VS THROUGHPUT ON GPU FOR VARIOUS METHODS

Method	Type	Acc (%)	Throughput
DeiT-small [7]	Flattened	79.9	940
ConViT [41]	Flattened	81.3	725
Siwn-Small [4]	Fixed Hierarchical	83.0	1720
FasterViT [18]	Fixed Hierarchical	84.2	3161
Paca [20]	Adaptive Hierarchical	84.0	1630
DualFormer [21]	Adaptive Hierarchical	84.8	2036
Sanger [22]	Sparse	79.2	1510
ReViT-w/o pruning	Adaptive Hierarchical	84.8	2036
ReViT	Adaptive Hierarchical & Sparse	84.9	3427

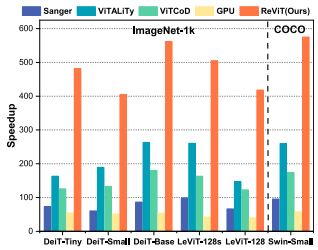

Fig. 14. Speedup over CPU.

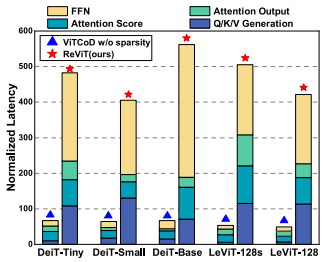

Fig. 15. Latency breakdown.

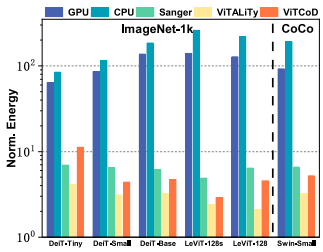

Fig. 16. Energy savings.

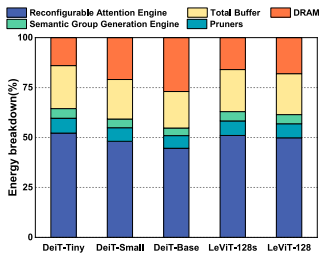

Fig. 17. Energy breakdown.

Algorithmic Optimization. Table IV shows the accuracy vs. throughput of the various methods on the GPU. We can observe that (i) ReViT outperforms all other baselines in terms of both accuracy and efficiency. Compared to vanilla flattened attention, ReViT-w/o pruning and ReViT achieve $2.1\times$ and $3.6\times$ times efficiency improvement, respectively. (ii) The efficiency of our ReViT-w/o pruning is lower than that of the most advanced FasterViT, while ReViT is superior to FasterViT after further introducing the sparse attention mechanism. This is because semantic-aware hierarchical attention introduces irregular attention computation, and the GPU cannot support the irregular computation well enough to translate the theoretical saving into practical performance improvement. Therefore, a specialized architecture is needed to release the potential of ReViT.

C. Architecture Performance

Speedup. Fig. 14 depicts the performance of our method compared with other baselines, including Sanger, ViTALiTy, ViTCoD, CPU, and GPU. On average, ReViT is $6.2\times$, $2.3\times$, $3.3\times$, $474.4\times$ and $9.9\times$ faster than Sanger, ViTALiTy, ViTCoD, CPU, and GPU respectively. First, in terms of algorithm design, our semantic-aware hierarchical differential attention mechanism exploits feature similarity within the same semantics to significantly reduce the computational workload and the number of DRAM accesses required for patch loading. In addition, our proposed semantic-aware hierarchical pruning further reduces the computational and communication costs by identifying and focusing on important foreground patches and excluding unnecessary computations in the background. Second, in terms of hardware design, the proposed *Reconfigurable Attention Engine* and the latency-aware out-of-order execution can flexibly allocate the computational resources to enhance the overall hardware utilization when processing the varying scales of hierarchical attention. Furthermore, the *Reconfigurable Bit-Serial Processing Element* can support end-to-end differential attention. It utilizes deltas for processing, storing, and communicating, greatly reducing the computational cost, storage cost, and communication overhead. The detailed latency breakdown compared with the standard attention is provided in Fig. 15.

Energy Consumption. Fig. 16 presents the normalized energy efficiency comparison between our method and other baseline accelerators. Overall, ReViT achieves energy savings of $6.2\times, 3.6\times, 5.0\times, 172\times, 111\times$ compared to Sanger, ViTALiTy, ViTCoD, CPU and GPU respectively. The performance gains come from multiple aspects: (i) The proposed *Reconfigurable Attention Engine* reduces the global scope of the information interaction to local semantic groups. Meanwhile, it utilizes deltas to perform end-to-end differential attention. (ii) Two *Pruners* effectively remove task-irrelevant backgrounds and patches. All these methods effectively reduce computation cost, DRAM access, and on-chip memory access. Fig. 17 illustrates the energy breakdown of ReViT across the typical benchmarks. On average, the DRAM, *Reconfigurable Attention Engine*, *Semantic Group Generation Engine*, *Pruners*, and the on-chip

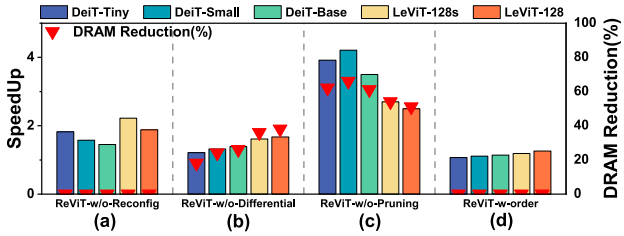

Fig. 18. Effectiveness of (a) semantic-aware attention, (b) semantic-aware differential attention, and (c) semantic-aware hierarchical pruning.

buffer account for 19.2%, 49.2%, 4.5%, 6.9% and 20.2% of the total energy consumption respectively. Over 60% of the total energy is consumed by the computation engines. The extra energy overhead (e.g., generating semantic groups and pruning) is negligible compared to the energy saved.

D. Ablation Study

Handling of Semantic-Aware Hierarchical Attention. To demonstrate the effectiveness of our co-designed architecture in meeting the adaptive hierarchical attention's irregular computation demands, we construct a variant ReViT-w /o-Reconfig that configures *Reconfigurable Attention Engine* as a single large execution engine. In this case, the irregular semantic groups could only be processed individually by *Reconfigurable Attention Engine*. Our experimental results in Fig. 18(a) indicate that the ReViT model outperforms ReViT-w/o-Reconfig by 1.8× in terms of speed. This improvement is primarily attributed to the abundance of smaller patches present within the dynamic hierarchical attention, which cannot fully occupy the entire PE array, thereby resulting in under-utilization of the PEs. Conversely, our *Reconfigurable Attention Engine* can flexibly allocate computational resources to patches with varied sizes to boost PE array usage. Furthermore, our latency-aware out-of-order execution flow mitigates idle times due to unbalanced workloads.

Effect of Semantic-Aware Differential Attention. To validate the effectiveness of semantic-aware differential attention, we compared our ReViT model against a variant, ReViT-w/o-Differential, which substitutes the computation matrix composed of the *Reconfigurable Bit-Serial Processing Elements* in ReViT with a traditional 64×64 systolic array. The comparison in Fig. 18(b) shows that the architecture with the RBSP achieves on average 1.4× speedup and 28% DRAM reduction compared to ReViT-w/o-Differential. It is mainly because our proposed RBSP can exploit feature similarity in the intra-group local attention, which greatly reduces the number of effectual bits required per value with delta execution. For instance, in our 8-bit quantization setup, the average number of effectual bits for DeiT-small is 3 for ImageNet-1k, and 3 for COCO. Furthermore, we can observe that differential attention provides a higher speedup for larger models, mainly because larger models potentially have more redundant features within local similarity clusters.

Effect of Semantic-Aware Hierarchical Pruning. To demonstrate the effectiveness of semantic-aware hierarchical pruning, we have compared our ReViT with its variant ReViT-w/o-Pruning, which refrains from employing the proposed pruning scheme. Our results in Fig. 18(c) indicate that ReViT is 3.4× faster and saves 58.8% DRAM access. From an algorithmic perspective, by utilizing local patch pruning and global group pruning, ReViT effectively reduces the number of irrelevant patches and foreground-background confusion, thereby reducing redundant computations. On a hardware level, the speed improvement is largely due to two lightweight pruning engines that share the same micro-architecture for efficient local patch pruning and global group pruning.

Effect of Latency-Aware Out-of-Order Execution Flow. To validate the effectiveness of latency-aware out-of-order execution flow, we have compared our ReViT with its variant ReViT-w-order, which employs sequential execution. The results in Fig. 18(d) indicate that ReViT is 1.15× faster than ReViT-w-order. Meanwhile, the hardware utilization of Revit is improved by 25%. This indicates that the proposed latency-aware out-of-order execution flow can minimize the waiting time for inter-group global attention and improve hardware utilization, thus enhancing the overall execution efficiency.

VI. Related Work

Due to the inefficiency of self-attention mechanisms, there has been an emergence of software and hardware co-designed transformer accelerators specialized for NLP tasks [22], [42], [43], [44], [45] and vision tasks [15], [16], [17]. These accelerators utilize dynamic sparse patterns to address the quadratic complexity of computing attention. For NLP tasks, SpAtten [42] structurally pruned unnecessary attention headers and input tokens. Sanger [22] utilized low-precision Q and K vectors to estimate sparse attention masks, and segmented with the support of reconfigurable architectures to make them more regular and efficient. DOTA [43] considered low-rank linear transformations to predict sparse attention masks and explores token-level parallelism and promiscuous execution for location-aware computation. For vision tasks, ViTCoD [15] pruned and polarized the attention map, and then coordinated both dense and sparse workloads to improve hardware utilization. HeatViT [16] reduced computation by pruning task-irrelevant patches. ViTALiTy [17] accelerates ViTs by unifying the low-rank and sparse components of attention. However, these approaches all use the flattened attention mechanism, ignoring the potential opportunity for speedup presented by the hierarchical semantics of images. In contrast, ReViT is the first algorithm-accelerator co-design framework dedicated to accelerating sparse ViT, leveraging the hierarchical semantics of images from both algorithmic and hardware perspectives.

VII. Conclusion

In this work, we reveal that hierarchical visual semantics is a fundamental property of real-world images. Inspired by this, we propose a systematic algorithm-architecture co-design

approach, called ReViT, which end-to-end explores the hierarchical semantics of images to accelerate ViT inference. Specifically, on the algorithm level, we propose a semantic-aware hierarchical differential attention mechanism, which exploits the same semantic class with strong feature similarity to reduce computation and communication in a differential attention mechanism, and support semantic-aware attention efficiently. On the hardware level, we design a fine-grained reconfigurable dedicated architecture to support the proposed algorithm and translate it into performance improvements. To the best of our knowledge, ReViT is the first systematic hardware-software co-design approach for exploring the underlying hierarchical semantics of images within the real world.

REFERENCES

[1] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6000–6010.

[2] S. Huang, Y. Liu, C. Fung, H. Wang, H. Yang, and Z. Luan, "Improving log-based anomaly detection by pre-training hierarchical transformers," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2656–2667, Sep. 2023.

[3] X. Zhou et al., "Personalized federation learning with model-contrastive learning for multi-modal user modeling in human-centric metaverse," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 4, pp. 817-831, Apr. 2024.

[4] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10012–10022.

[5] C. H. Song, J. Yoon, S. Choi, and Y. Avrithis, "Boosting vision transformers for image retrieval," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2023, pp. 107–117.

[6] B. Graham et al., "LeViT: A vision transformer in convnet's clothing for faster inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12259–12269.

[7] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 10347–10357.

[8] Z. Dai, B. Cai, Y. Lin, and J. Chen, "UP-DETR: Unsupervised pre-training for object detection with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1601–1610.

[9] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[10] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Comput. Surveys*, vol. 54, no. 10s, pp. 1–41, 2022.

[11] Z. Kong et al., "SpViT: Enabling faster vision transformers via latency-aware soft token pruning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Tel Aviv, Israel, Part XI. Berlin, Heidelberg, Germany: Springer, 2022, pp. 620–640.

[12] M. Marszalek and C. Schmid, "Semantic hierarchies for visual object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Piscataway, NJ, USA: IEEE Press, 2007, pp. 1–7.

[13] T. Wang et al., "VisualNet: An end-to-end human visual system inspired framework to reduce inference latency of deep neural networks," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2717–2727, Nov. 2022.

[14] N. Kruger et al., "Deep hierarchies in the primate visual cortex: What can we learn for computer vision?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1871, Aug. 2013.

[15] H. You et al., "ViTCoD: Vision transformer acceleration via dedicated algorithm and accelerator co-design," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 273–286.

[16] P. Dong et al., "HeatVit: Hardware-efficient adaptive token pruning for vision transformers," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 442–455.

[17] J. Dass et al., "Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear Taylor attention," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 415–428.

[18] A. Hatamizadeh et al., "FasterViT: Fast vision transformers with hierarchical attention," 2023, *arXiv:2306.06189*.

[19] W. Zeng et al., "Not all tokens are equal: Human-centric visual analysis via token clustering transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11101–11111.

[20] R. Grainger, T. Paniagua, X. Song, N. Cuntoor, M. W. Lee, and T. Wu, "PaCa-ViT: Learning patch-to-cluster attention in vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18568–18578.

[21] Z. Jiang, L. Liu, J. Zhang, Y. Wang, M. Chen, and C. Wang, "Dual path transformer with partition attention," 2023, *arXiv:2305.14768*.

[22] L. Lu et al., "Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO-54)*, 2021, pp. 977–991.

[23] K. Choromanski et al., "Rethinking attention with performers," 2020, *arXiv:2009.14794*.

[24] C. Zhou, Y. Zhang, J. Chen, and D. Huang, "OcTr: Octree-based transformer for 3d object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 5166–5175.

[25] Y. Xu et al., "Evo-ViT: Slow-fast token evolution for dynamic vision transformer," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 3, pp. 2964–2972.

[26] A. Dasgupta, R. Kumar, and T. Sarlós, "Fast locality-sensitive hashing," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1073–1081.

[27] M. Mahmoud, K. Siu, and A. Moshovos, "Diffy: A Déjà vu-free differential deep neural network accelerator," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 134–147.

[28] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[29] X. Zeng et al., "Addressing irregularity in sparse neural networks through a cooperative software/hardware approach," *IEEE Trans. Comput.*, vol. 69, no. 7, pp. 968–985, Jul. 2020.

[30] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–12.

[31] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "DynamicViT: Efficient vision transformers with dynamic token sparsification," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 13937–13949.

[32] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, and Y. Guo, "Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18953–18962.

[33] J. Albericio et al., "Bit-pragmatic deep neural network computing," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2017, pp. 382–394.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Piscataway, NJ, USA: IEEE Press, 2009, pp. 248–255.

[35] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, Zurich, Switzerland, Part V 13. Berlin, Heidelberg, Germany: Springer, 2014, pp. 740–755.

[36] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, "PyTorch," in *Programming With TensorFlow: Solution for Edge Computing Applications*, Berlin, Heidelberg, Germany: Springer, 2021, pp. 87–104.

[37] Z. Li and Q. Gu, "I-ViT: Integer-only quantization for efficient vision transformer inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 17065–17075.

[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–18.

[39] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible DRAM simulator," *IEEE Comput. Archit. Lett.*, vol. 15, no. 1, pp. 45–49, Jan.–Jun. 2016.

[40] S. Thoziyoor, N. Muralimanohar, J. Ahn, and N. P. Jouppi, "Cacti 5.1," Tech. Rep. HPL-2008-20, HP Labs, 2008, pp. 1–37.

[41] S. d'Ascoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, and L. Sagun, "ConViT: Improving vision transformers with soft convolutional inductive biases," in *Proc. Int. Conf. Mach. Learn.*. PMLR, 2021, pp. 2286–2296.

[42] H. Wang, Z. Zhang, and S. Han, "SpAtten: Efficient sparse attention architecture with cascade token and head pruning," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 97–110.

[43] Z. Qu, L. Liu, F. Tu, Z. Chen, Y. Ding, and Y. Xie, "DOTA: Detect and omit weak attentions for scalable transformer acceleration," in *Proc. 27th ACM Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2022, pp. 14–26.

[44] Z. Li, S. Ghodrati, A. Yazdanbakhsh, H. Esmaeilzadeh, and M. Kang, "Accelerating attention through gradient-based learned runtime pruning," in *Proc. 49th Annu. Int. Symp. Comput. Archit.*, 2022, pp. 902–915.
[45] H. Wang, H. Xu, Y. Wang, and Y. Han, "CTA: Hardware-software co-design for compressed token attention mechanism," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 429–441.

Xiaofeng Zou received the Ph.D. degree in computer science and technology from Hunan University, China, in 2023. His research interests include parallel computing, computer architecture, efficient machine learning, and deep learning. He has published several research articles in international conferences and journals, such as *MICRO*, *HPCA*, *DAC*, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Cen Chen (Senior Member, IEEE) received the Ph.D. degree in computer science from Hunan University, China. Currently, he is a Professor with the School of Future Technology, South China University of Technology, China. His research interests include parallel and distributed computing, computer architecture, machine learning, and deep learning. He has published several research articles in international conferences and journals on machine learning algorithms and parallel computing, such as *MICRO*, *HPCA*, *DAC*, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, AAAI, IEEE International Conference on Data Mining, ICPP, ICDCS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE TRANSACTIONS ON CYBERNETICS. He is also an Associate Editor of IEEE TRANSACTIONS ON COMPUTERS.

Hongen Shao is currently working toward the Ph.D. degree with the School of Future Technology, South China University of Technology, China. His research interests include artificial intelligence algorithms and architecture design.

Qinyu Wang is currently working toward the Ph.D. degree with the School of Future Technology, South China University of Technology, China. His research interests include parallel and distributed computing, intelligent computing, and deep learning.

Xiaobin Zhuang is currently working toward the master's degree with the School of Future Technology, South China University of Technology. His research interests include deep learning and parallel computing.

Yangfan Li received the bachelor degree in engineering from the School of Automation, Huazhong University of Science and Technology, in 2015, and the Ph.D. degree in computer science from Hunan University, China, in 2022. Currently, he is a Lecturer with the School of Computer Science and Engineering, Central South University, China. His research interests include computer architecture, efficient machine learning, and deep learning.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of computer science with the State University of New York and a National Distinguished Professor with Hunan University, China. His research interests include cloud computing, fog computing and mobile edge computing, high-performance computing, computer architectures and systems, and intelligent and soft computing. He has authored or co-authored more than 990 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently an Associate Editor of *ACM Computing Surveys*. He has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is an AAIA fellow and a member of Academia Europaea (Academician of the Academy of Europe).