Hybrid Token Selector based Accelerator for ViTs

Akshansh Yadav*
Department of CSE
Indian Institute of Technology Jodhpur
Jodhpur, India
d23cse001@iitj.ac.in

Anadi Goyal*
Department of CSE
Indian Institute of Technology Jodhpur
Jodhpur, India
goyal.24@iitj.ac.in

Palash Das
Department of CSE
Indian Institute of Technology Jodhpur
Jodhpur, India
palashdas@iitj.ac.in

***Abstract*—Vision Transformers (ViTs) have shown great success in computer vision but suffer from high computational complexity due to the quadratic growth in the number of tokens processed. Token selection/pruning has emerged as a promising solution; however, early methods introduce significant overhead and complexity. Applying a token selector in the early layers of a ViT can yield substantial computational savings (GFLOPs) compared to using it in later layers. However, this approach often leads to significant accuracy loss, particularly with the popular Attention-based Token Selection (ATS) technique. To address these issues, we propose a hybrid token selection (HTS) strategy that integrates our Keypoint-based Token Selection (KTS) with the existing ATS method. KTS dynamically selects important tokens based on image content in the early layers, while ATS refines token pruning in the later layers. This hybrid approach reduces computational costs while maintaining accuracy. Additionally, we design custom hardware modules to accelerate the execution of the proposed methods and the ViT backbone. The proposed HTS delivers a 35.85% reduction in execution time relative to the baseline without any token selection. Furthermore, our results demonstrate that HTS achieves up to a 0.39% increase in accuracy and offers up to 6.05% savings in GFLOPs compared to existing method.**

***Index Terms*—Vision Transformer Accelerator; Token Pruning;**

I. Introduction

Vision Transformers (ViTs) [3] are widely used in image recognition, object detection, and segmentation. However, they struggle with efficiency, especially in resource-constrained environments, due to the quadratic growth in computational complexity as the number of tokens [3] increases. Token pruning has emerged as a promising solution to reduce the number of tokens processed by the models, thereby directly targeting the source of this quadratic complexity. Early token pruning methods [2], [11], [12] have used neural networks for token selection, effectively reducing computations (GFLOPs) during inference. However, these methods introduce high overheads due to the additional scoring network and complicate training. Recent work [6] addresses this issue using attention-based token selection (ATS), which leverages precomputed attention scores from ViT's multi-head self-attention (MSA) layers to evaluate token importance. While ATS is promising, it introduces challenges detailed in Issue 1. We review the state-of-the-art and identify another challenge, outlined in Issue 2.

Issue 1 ***(a)*** *Initiating pruning with ATS in the early layers of ViTs significantly reduces GFLOPs but often results in high accuracy loss.* ***(b)*** *A more accuracy-friendly approach is to apply ATS across multiple layers with a gradually increasing pruning rate, which helps minimize accuracy loss with decent savings in GFLOPs. However, it involves overhead of multiple token selectors, particularly for the deeper ViT models.*

[*]Both the authors have contributed equally to this work.

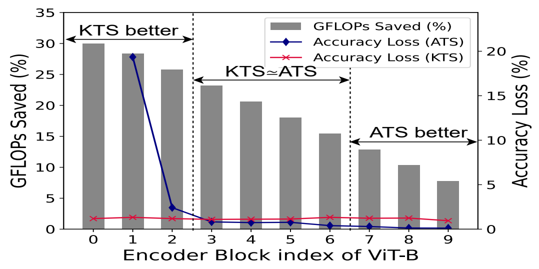


Fig. 1: Trade-off between Savings and Accuracy Loss.

Figure 1 manifests the claim of Issue 1 (a), with each bar showing the GFLOPs savings from pruning tokens after the respective encoder blocks of ViT-Base. We can notice that pruning tokens after the 9th block yields less savings than pruning after 2nd block. Both proposed keypoint token selection (KTS) and ATS are configured to remove same number of tokens in each layer, ensuring equal savings as shown in Figure 1. However, ATS pruning in early layers results in significant accuracy loss (shown in Figure 1), whereas KTS performs better. ATS excels in later layers, and both methods perform almost similarly in the middle layers. The reason ATS is not well-suited for early ViT layers can be visualized in

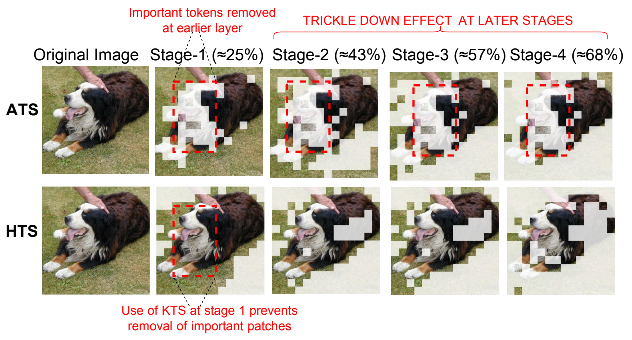

Important tokens removed at earlier layer
TRICKLE DOWN EFFECT AT LATER STAGES
Original Image
Stage-1 (≈25%)
Stage-2 (≈43%)
Stage-3 (≈57%)
Stage-4 (≈68%)
ATS
HTS
Use of KTS at stage 1 prevents removal of important patches

Fig. 2: Comparison of token selection in ATS and HTS.

978-3-9826741-0-0/DATE25/© 2025 EDAA

Figure 2. Due to immature or noisy attention scores in ATS, tokens with important features are removed in the early stages, causing a trickle-down effect in the later layers that significantly harms the accuracy of ViTs. In contrast, KTS removes tokens with non-essential features in the early stages, resulting in lesser trickle-down effects in later layers and minimal loss in accuracy. This observation motivates us to explore a hybrid approach that combines KTS for early layers and ATS for later layers, providing a balanced computational savings and minimal accuracy loss.

Issue 2 *Another limitation of existing methods is their reliance on static token selection [6], [8], [9], which prunes a fixed number of tokens regardless of image content. This approach can either over-prune, reducing accuracy, or under-prune, missing potential performance gains.*

Towards resolving these issues, we design a Keypoint-based token selector (KTS), identify its optimal placement among the ViT layers, and propose to use it by handshaking with the existing ATS technique. Our contributions are as follows.

1) **Keypoint-based token selection (KTS)** We introduce KTS to address Issues 1(a) and 2. KTS dynamically identifies critical regions using FAST keypoint detection [13], scores tokens based on keypoint (informative pixels) density in the corresponding image patches, and prunes tokens below a threshold. As shown in Figure 1 and 2, KTS is highly effective, minimizing accuracy loss compared to ATS when applied to the early blocks (Block 0-2) while maintaining similar computational efficiency. Additionally, KTS addresses Issue 2 by dynamically adjusting the number of retained tokens based on each image's content.
2) **Circle-based Keypoint Scoring** We enhance the efficiency of KTS by incorporating a circle-based keypoint scoring technique. Rather than simply scoring patches by the number of keypoints labeled by FAST, which might overlook important features, we introduced a method that draws a fixed-radius circle around each keypoint. This approach selects additional pixels as keypoints and scores patches based on all keypoints within these circles. This ensures that critical tokens are never missed, leading to improved classification accuracy.
3) **Hybrid token selection** We propose a Hybrid Token Selector (HTS) that strategically combines both KTS and ATS. Since KTS is more effective in the early layers in terms of accuracy loss, we apply it first, followed by ATS with cumulative pruning across subsequent blocks. This approach mitigates the overhead associated with ATS, addressing Issue 1 (b), as KTS minimizes the number of tokens that ATS has to repeatedly process in subsequent layers, enhancing overall efficiency. By applying KTS in early blocks, HTS improves accuracy while maintaining equal or better computational savings compared to using ATS or KTS alone.
4) **Custom Hardware Design** We design custom hardware for KTS, ATS, and the combined HTS, along with hardware modules for ViT backbone. We also introduce a dataflow that enables parallelism in token selection and inference, ensuring high system throughput.

II. Background and Related Work

A. *Token selection in Vision Transformer*

ViTs [3] process images by creating patch embeddings, which are passed through encoder blocks with Multi-head Self Attention (MSA), Multi-Layer Perceptron (MLP), and LayerNorm (LN), ultimately leading to the classification head. The details of ViT are provided in previous work [3]. Note that a patch refers to a portion of the input image, while a token is the corresponding embedding of that patch used in ViT inference. Token selection/pruning in ViTs can be categorized into two main approaches: static and dynamic. Static token selection uniformly removes tokens in an input-agnostic manner, which can sometimes eliminate crucial tokens. Techniques in this category use attention scores between the CLS token and others to determine which tokens to retain [6], [8], [9]. Others use a trained token selector to determine keep probabilities for each token, ensuring important tokens are retained [5]. Dynamic token selection retains informative tokens based on each input image. This adaptive approach can be more effective at maintaining essential tokens, using techniques like sampling methods [4], [17] or reinforcement learning [11]. These methods include high computational overhead but can be beneficial for real-time performance. Both static and dynamic token selections lead to faster execution of ViTs. In addition to these token selectors, another approach introduces Graph-based Token Propagation (GTP) [16], which utilizes attention scores and reduces computational complexity while maintaining the essential information of eliminated tokens. Token selection can also be divided into single-stage, where pruning occurs once, and multi-stage, where token selection is applied at various stages where each stage can consists of more than one encoder blocks. A faster execution can even be achieved using specialized hardware for ViTs, as discussed in the next section.

B. *Vision Transformer Accelerator*

To accelerate ViTs, VAQF [14] and Auto-ViT-Acc [7] use low-precision quantization. VAQF adjusts activation precision based on frame rate, while Auto-ViT-Acc quantizes weights and activations using fixed-point and power-of-two (PoT) ratios. ViA [15] improves computation and memory efficiency through a partitioning strategy, and ViTA [10] addresses memory constraints using column-level weight matrix scheduling. However, these works do not utilize token selection to exploit layer redundancy in ViTs. In contrast, HeatViT [2] and ViT-ToGo [6] integrate token pruning to enhance hardware efficiency. HeatViT employs attention-based token selection to dynamically remove non-informative tokens, optimizing ViT performance on FPGAs. ViT-ToGo implements grouped token pruning with a head-wise importance estimator designed for FPGA implementation. The comparison with these works is explained in the subsequent section.

Algorithm 1: Proposed Token Pruning Technique

```
1  Function KTS(image, r, θ): // Keypoint-based Token Selector
       Input: RGB image, radius $r$, threshold $\theta$
       Output: Indices of important tokens $P$
2      $grayscale \leftarrow 0.299 * R + 0.587 * G + 0.144 * B$;
3      $keypoints \leftarrow$ FAST($grayscale$) ; // Detect Keypoints
4      for each $(x, y) \in keypoints$ do
5          Mark all pixels within radius $r$ around $(x, y)$ as important
6      for each patch $p_i$ do
7          $score_i \leftarrow$ count of important pixels in patch $p_i$
8      $P \leftarrow \{i \mid score_i \geq \theta\}$ ; // Select Important Tokens
9      return $P$;
10 Function ATS(A, α): // Attention-based Token Selector
       Input: CLS attention scores $A \in \mathbb{R}^{n \times H}$, pruning rate $\alpha$
       Output: Indices of important tokens $P$
11     $a \leftarrow \text{mean}_h(A)$ ;          // Avg attention across heads
12     [sorted_indices] $\leftarrow$ sort($a$) ; // descending sorting
13     $k \leftarrow \lfloor (1 - \alpha) \times n \rfloor$ ; // Tokens to keep
14     $P \leftarrow$ sorted_indices$[0 : k]$ ; // Top-k tokens
15     return $P$;
16 Function HTS(α, B, RGB Image): // Hybrid Token Selector
       Input: Pruning rate $\alpha$, block indices for token removal $B = [b1, b2, b3]$,
              RGB Image
       Output: Final processed output
17     $indices\_to\_keep \leftarrow$ KTS(RGB image) ;    // Token selection
        using KTS
18     for each block $i$ from 1 to total number of blocks do
19         $attention, input_2 \leftarrow$ MSA($input[indices\_to\_keep]$);
20         if $i \in B$ then
21             $indices\_to\_keep \leftarrow$ ATS($attention, \alpha$) ; // ATS
22             $output \leftarrow$ MLP($input_2$);
23             $output \leftarrow$ output$[indices\_to\_keep]$;
24         else
25             $output \leftarrow$ MLP($input_2$) ; // No pruning
26     return $output$;
```

III. Proposed Token Pruning Techniques

A. Proposed KTS Algorithm

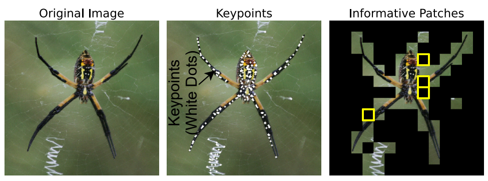
As discussed in Section I, the attention score technique struggles with accuracy in early layers, though it provides greater computational savings. To mitigate this issue, we propose Keypoint-based Token Selector (KTS), a dynamic token selection method that avoids relying on attention scores. Note that in KTS, information from image patches is used to identify non-informative patches, and the tokens representing the embeddings of these patches are removed during ViT inference. The core concept of our algorithm is to efficiently pinpoint regions of interest in an image, particularly where the main object is located. Leveraging this information, the algorithm categorizes tokens by their significance, retaining only the informative ones for inference. To identify the regions, we use the FAST algorithm [13] to detect keypoints, which represent the critical pixels of an image. FAST functions as a corner detection method, identifying points where pixel intensity changes sharply in various directions. The KTS algorithm begins with FAST and incorporates additional steps outlined in Algorithm 1, specifically in lines 1 to 9. KTS converts the input RGB image into grayscale using the formula mentioned in line 2. The FAST function is then applied on the grayscale image, yielding a list of coordinates of the keypoint pixels (line 3). Using these coordinates as centers, KTS draws circles with radius r, further designating the pixels within these circles as keypoints (lines 4-5). The rationale for using circles is explained in the succeeding sections. Following this, each keypoints of the image is mapped to its corresponding patch. We then assign a score to each patch based on the count of keypoints within it (lines 6-7). Essentially, a higher number of keypoints results in a higher score for the respective patches. Finally, patches with scores above an experimentally determined token_score_threshold (θ) are retained, and their indices are stored in a list P, which is returned (lines 8-9) to the encoder blocks to prune non-informative tokens, reducing computations.

B. Proposed Hybrid Token Selection (HTS)

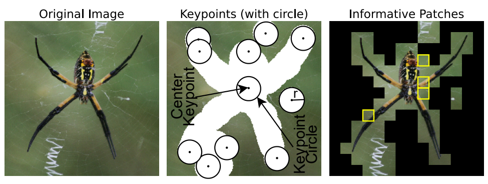
Figures 1 and 2 have demonstrated the need for KTS in the early layers and ATS in the later layers of a ViT. Towards that direction, we propose a hybrid token selector (HTS) that combines the strengths of KTS and ATS. ATS works by averaging the attention scores across heads (line 11), sorting the tokens based on these averaged scores (line 12), and then selecting the top-k tokens to keep according to the pruning rate α (lines 13-15). For instance, if α is set to 0.2, only the top 80% of tokens with the highest scores are retained. The operations of the proposed HTS are as follows. It initially uses KTS to select tokens based on keypoint detection (line 17). For each encoder block, attention scores are then computed, and if the block is within the designated pruning stages (B), ATS is applied to further refine token selection (lines 18-21). After the MLP processing (line 22), the resulting output is updated to keep only the informative tokens (line 23), while blocks outside of B process all tokens without additional pruning (lines 24-25).

C. Optimizing Token Selection in KTS

Our approach utilizes keypoints to identify important regions in the image, enabling it for the dynamic selection of crucial tokens based on the image content. To assign a score to each patch based on the extracted keypoints, we explore two methods: ① Direct Keypoint Scoring and ② Circle-based Keypoint Scoring. In direct keypoint scoring, each patch is assigned a score based on the number of keypoints identified by the FAST algorithm. Although this method is straightforward,



(a) Direct Keypoint Scoring.



(b) Circle-based Keypoint Scoring.

Fig. 3: Direct Keypoint Vs. Circle-based Keypoint Scoring.

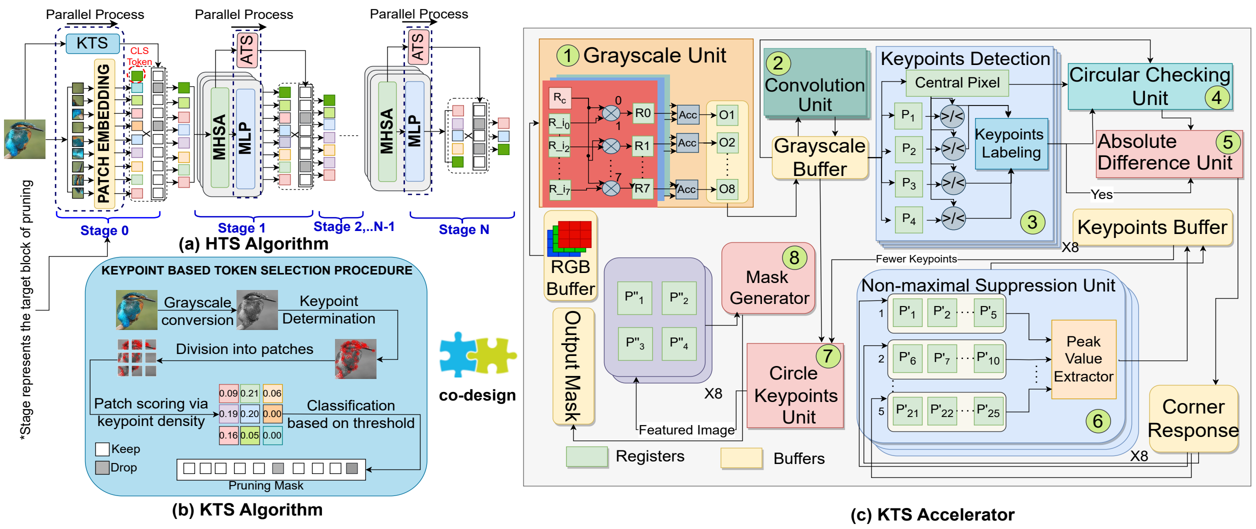


Fig. 4: The Proposed HTS Architecture

it has a limitation. As illustrated in Figure 3a, this method can miss critical patches with important features, highlighted in yellow squares, which may result in misclassification. To resolve this issue, we introduce a circle-based keypoint scoring mechanism. This technique involves drawing a circle of experimentally determined radius around each keypoint detected by the FAST algorithm, as shown in Figure 3b. The choice of a circle was based on the fact that circles cover an isotropic region, uniformly capturing the local neighborhood around keypoints without introducing directional bias, unlike shapes such as squares or ellipses. Pixels within each circle are also considered important and labeled as keypoints, leading to increased keypoints for an input image. Each patch is then scored based on the number of all keypoints it contains, ensuring that crucial patches receive scores above the token_score_threshold (θ) and are retained. This circle-based keypoint scoring method overcomes the limitations of the direct keypoint approach by preserving essential features, thereby reducing the risk of misclassification. This optimization increases the accuracy by around 4-10% across the ViT models on ImageNet [1], compared to direct keypoint scoring, which is highly aggressive in removing important tokens.

IV. Proposed Architecture

By integrating KTS into the early layers, HTS offers twofold benefits: (1) reducing the computational overhead in each encoder block, and (2) lowering the computational demands in ATS for later layers. While designing HTS, we strategically position KTS and ATS to overlap their execution time with the critical path of the ViT backbone, as shown in Figure 4a. As KTS is applied early, this can be overlapped with the patch embedding layer. Since ATS depends on the attention scores from the MSA module and its results are only required after the MLP, ATS can safely run concurrently with MLP execution. This overlap is feasible because the MLP is more time-consuming than ATS. We have designed custom hardware for each module to enable end-to-end processing. With KTS being the core module of our HTS algorithm, we present its detailed steps in Figure 4b, while the corresponding hardware co-design is illustrated in Figure 4c. The KTS unit is composed of eight primary components: (1) Grayscale Conversion unit, (2) Convolution unit, (3) Keypoint Detection unit, (4) Circular Checking unit, (5) Absolute Difference unit, (6) Non-Maximal Suppression unit, (7) Circle Keypoint unit, and (8) Mask Generator. The computational focus of these modules is on keypoint detection and non-maximal suppression, where each pixel is analyzed as a potential keypoint. To minimize off-chip access, the RGB image is loaded into an accelerator buffer. In the first stage, RGB is converted to grayscale by processing each color channel separately in module 1 (shown in Figure 4 (c)), using three parallel multiplier stacks, each containing 8 units. The results are combined in an accumulator array to produce grayscale pixels, which are stored in output registers and transferred to the grayscale buffer. After convolution in module 2, the image is returned to the grayscale buffer for further processing. Keypoint detection is then performed on the convolved image using two methods: FAST (module 3) and circular keypoint detection (module 4). FAST identifies potential keypoints by comparing the intensity of a central pixel with four neighboring pixels in a cross pattern. If at least three neighboring pixels have significantly higher or lower intensity, the central pixel is marked as a keypoint. To enhance efficiency, two levels of parallelization are implemented: (1) the four intensity comparisons are conducted simultaneously using four comparators, and (2) multiple pixels are processed concurrently using a stack of register-comparator modules. Pixels identified as keypoints are transferred to the keypoints buffer. If a pixel is not designated as a keypoint in module 3, it is further analyzed in the circular checking unit (module 4) to determine weather it is a keypoint. This time, the central pixel is compared with 16 surrounding pixels. To maintain the same level of parallelism, we increase the hardware resources accordingly. Once a keypoint is identified, the Absolute Difference unit (module 5) updates its pixel value by summing the absolute differences between the keypoint and its neighboring pixels used in circular checking. These updated values are then sent to the corner image buffer for further processing in the Non-Maximal Suppression Unit (NMS) (module 6). After keypoint detection across the entire image, NMS refines the results by selecting the highest pixel value in each 5×5 window while suppressing nearby pixels. As shown in module 6, the peak value extractor module processes five pixels simultaneously, with eight such windows handled concurrently. Next, the circle keypoint unit (module 7) uses a circle-based scoring method to increase keypoints. The mask generator (module 8) then assigns scores to the patches based on the detected keypoints, selecting the most informative patches by comparing their scores against an experimentally determined threshold, while suppressing those that fall below this threshold. Finally, a mask identifying the patches or tokens to retain is generated and sent to the output mask buffer. We have also developed specialized hardware modules for the ATS and the ViT backbone. The ViT backbone includes modules for Patch Embedding, LayerNorm, Multi-Head Self-Attention (MSA), MLP, and the Classifier Head. The dominant computation across these modules involves matrix multiplication, in most cases with bias addition. Our approach utilizes a block-based strategy where input matrices (weights, where applicable) are divided into smaller blocks. Each block is sequentially loaded into accelerator buffer, followed by the corresponding computation—whether it is matrix multiplication with or without bias addition, or both. The resulting block is then stored in an output buffer. This process is pipelined to maximize parallelism: while one block is being processed, the next block is loaded into the buffer. We further optimize the Attention Token Selector (ATS) module. First, average attention scores for all tokens are computed in parallel across multiple heads. Then, a hardware-optimized sorting mechanism ranks the tokens based on these scores. Finally, the top K tokens are selected for the next encoder block.

V. Experimental Evaluation

A. Experimental Setup

To validate our KTS and HTS method with the ViT models, we conduct experiments on ViT-Large, Base, Small and Tiny models using the ImageNet dataset. For hardware implementation, we use Vitis HLS (High-Level Synthesis) and Vivado, targeting the ZCU104 board, which includes 624 BRAM18k blocks, 1728 DSPs, 460.8k flip-flops (FFs), and 230.4k lookup tables (LUTs). The hardware is implemented for the ViT-Tiny model, running at a frequency of 500 MHz with 16-bit fixed-point (FX16) and 8-bit fixed-point (FX8) precision. In KTS, tokens are removed before the first encoder block, while in ATS, they are discarded after the 3^{rd}, 6^{th}, and 9^{th} blocks in ViT models. In ViT-Large, for ATS, tokens are removed after the 6^{th}, 12^{th}, and 18^{th} blocks to match its deeper architecture and maintain a consistent patch drop rate with minimal accuracy loss. Additionally, KTS token selection depends on two hyper-parameters: the token_score_threshold 'θ' and the radius 'r' for the circle-based keypoint scoring method. This hyper-parameters are reconfigurable and can be adjusted as an when required. We determine 'θ' based on the average number of tokens to be discarded across all images in a dataset, which can be adjusted according to the acceptable loss in accuracy for an application. Table I presents the average number of tokens dropped from the ImageNet dataset at different values of θ. This reduction is achievable in stage 0 of a ViT by virtue of KTS, leading to savings in the subsequent encoder blocks and ATS modules. The optimal value of 'r' is experimentally determined as 18 for the mentioned model dataset pairs.

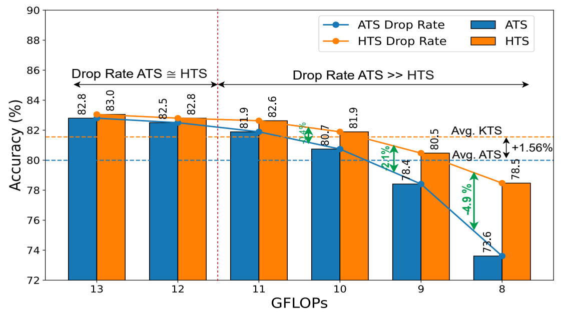


Fig. 5: Accuracy Trends in ATS vs. HTS across GFLOPs.

Token Score Threshold (θ)	Tokens Pruned (%)	Token Score Threshold (θ)	Tokens Pruned (%)
0	0	50	24.49
10	17.97	60	26.26
20	19.69	70	28.24
30	21.20	80	30.45
40	22.84	90	33.27

TABLE I: Token Pruned (%) for the corresponding θ.

B. Results

*1) **Accuracy Vs. GFLOPs**:* Figure 5 compares the accuracy and computational cost (GFLOPs) of ViT-base using ATS (multi-stage) and our HTS, with an equal number of tokens removed using both methods. As token pruning reduces computations (GFLOPs on the X-axis), accuracy (Y-axis) declines for both ATS and HTS. However, ATS shows a sharper accuracy drop compared to our HTS. HTS consistently maintains slightly higher accuracy across all GFLOP levels, with the accuracy gap between ATS and HTS widening as GFLOPs decrease. For instance, ATS accuracy drops by 9.19% from 13 to 8 GFLOPs, while HTS only decreases by 4.58%. This analysis indicates

ViT-Small			
Techniques	**Pruning Rates (Block 0/3/6/9)**	**Accuracy (%)**	**GFLOPs**
Baseline	0/0/0/0	78.11	4.53
ATS (Single)	0.3/0/0/0	55.57 (-22.54)	3.22 (-28.91 %)
KTS (Single)	0.3/0/0/0	74.63 (-3.48)	3.10 (-31.56 %)
ATS (Multi)	0/0.27/0.27/0.27	74.92 (-3.19)	2.94 (-35.09 %)
HTS (Multi)	0.15/0.15/0.15/0.15	76.13 (-1.98)	2.94 (-35.09 %)
ViT-Large			
Techniques	**Pruning Rates (Block 0/6/12/18)**	**Accuracy (%)**	**GFLOPs**
Baseline	0/0/0/0	84.65	61.15
ATS (Single)	0.3/0/0/0	71.73 (-12.87)	43.14 (-29.45%)
KTS (Single)	0.3/0/0/0	82.40 (-2.25)	42.36 (-30.73%)
ATS (Multi)	0/0.3/0.3/0.3	82.86 (-1.74)	38.23 (-37.49%)
HTS (Multi)	0.15/0.25/0.25/0.25	83.08 (-1.57)	37.20 (-39.16%)

TABLE II: Comparing token selectors at various pruning rates.

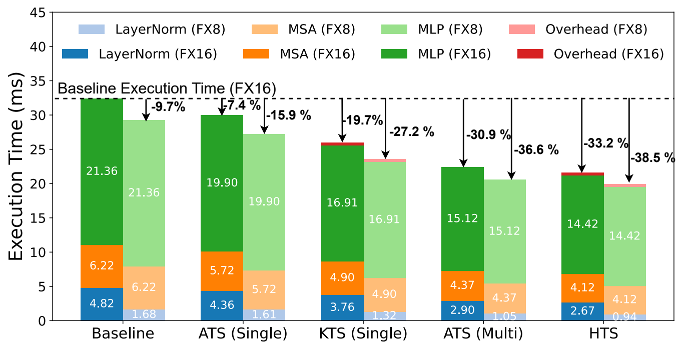


Fig. 6: Execution Time using various token pruning methods.

Techniques	Pruning rate	Accuracy drop	GFLOPs Savings
ViT-ToGo [6]	0/0.20/0.20/20	0.45%	26.83%
HTS (ours)	0.15/0.11/0.11/0.11	**0.54% (+0.09%↑)**	**31.76% (+4.93%↑)**
ViT-ToGo [6]	0/0.25/0.25/25	0.99%	31.89%
HTS (ours)	0.15/0.17/0.17/0.17	**0.71% (-0.28%↓)**	**37.60% (+5.71%↑)**
ViT-ToGo [6]	0/0.30/0.30/30	1.84%	36.59%
HTS (ours)	0.15/0.22/0.22/0.22	**1.45% (-0.39%↓)**	**42.64% (+6.05%↑)**

TABLE III: Comparison with ViT-ToGo [6].

that HTS is more robust in maintaining accuracy, while ATS is more sensitive to reduced computational costs due to token pruning. Table II presents the GFLOPs and corresponding accuracy for various token selectors at different pruning rates for ViT-Small and ViT-Large. The pruning rates, shown in the second column, represent the percentage of patches removed after specific blocks during ViT inference. For example, in HTS, 0.15/0.15/0.15/0.15 indicates 15% token pruning for KTS before the first layer (block 0) and 15% pruning for ATS after blocks 3, 6, and 9 in ViT-Small.

*2) **Performance Analysis**:* In Figure 6, the execution times of the ViT-tiny on the ZCU104 FPGA are compared using various token pruning techniques at FX16 and FX8 precisions. We compare the baseline (no token pruning) with two attention-based approaches—ATS (Single) and ATS (Multi)—as well as the two proposed methods, KTS and HTS. We adjust the pruning rates to reduce GFLOPs to a point that delivers similar model accuracy. This ensures a fair comparison of execution time with different token selectors. All pruning techniques significantly reduce execution time compared to the baseline (no pruning). In single-stage pruning, the proposed KTS (Single) achieves an average of 11.8% more savings than ATS (Single) over the baseline (FX16), averaged across both precisions. Multi-stage pruning with ATS outperforms KTS, reducing execution time by an additional 10.3% (on average). The proposed HTS, combining both KTS and ATS, delivers the best performance among all, achieving an overall reduction of 33.2% and 38.5% in execution time for FX16 and FX8 respectively, compared to baseline (FX16). Figure 6 also shows the time distribution across the main layers of the ViT, including the overhead introduced by the token selector modules. Both KTS and HTS incur minimal overhead, contributing only 1.65% and 4.6% to the total execution time, respectively, while delivering significant performance improvements.

Resource (Available)	Resource Utilization (%)			
	Baseline	HTS	ViT-ToGo [6]	HeatViT [2]
BRAM18K (624)	62%	62%	83.81%	355.5 (BRAM36K)
DSP (1728)	33%	38%	11.28%	1968
FF (460800)	20%	27%	9.40%	126k
LUT (230400)	25%	37%	18.01%	137.5k
Power	4.31W	4.6W	4.057W	9.453W

TABLE IV: Comparison of the Resource Utilization.

VI. Comparison with Earlier Works

Hardware implementations of token pruning-based ViT accelerators are still limited, but two notable works, ViT-ToGo [6] and HeatViT [2], have made significant strides with their FPGA-based hardware implementations of token pruning techniques for ViT. ViT-ToGo selects important tokens using attention scores, grouping adjacent tokens and pruning them based on collective importance. However, as discussed in Section I, this attention-based method is less suitable for the early layers, as the reliance on attention scores can lead to a significant loss of accuracy. Additionally, ViT-ToGo employs a static, input-agnostic token selector. In contrast, HeatViT introduces a dynamic token selection method that indirectly utilizes attention scores and refines the process with an MLP-based token selector to determine final token importance. HeatViT further enhances the performance through well-known optimizations such as quantization and polynomial approximation in its hardware implementation. Our proposed HTS addresses the limitations of ViT-ToGo by implementing a keypoint-based dynamic selection for the early layers of ViT, leading to additional computational savings. HTS outperforms ViT-ToGo in GFLOPs savings and accuracy in most cases, as shown in Table III for the ViT-base model on ImageNet. Although HeatViT combines multiple techniques for aggressive performance gains, it consumes 1.8x more BRAM and 3x more DSPs, resulting in roughly 2x higher power consumption compared to HTS. The quantitative comparison of resource utilization for the ViT-tiny model is presented in Table IV.

VII. Conclusion

This work addresses the limitations of attention-based token selectors (ATS) by proposing KTS, which removes tokens in early ViT layers using input image information. In later layers, ATS is applied, creating a hybrid solution (HTS) that is computationally efficient with minimal accuracy loss. We also design dedicated hardware units for KTS, ATS, and the ViT backbone, each equipped with parallel processing capabilities. Additionally, the token selectors are strategically placed to reduce their overhead by overlapping their execution time with the primary ViT execution. The cumulative outcome maximizes the system's throughput. The proposed system outperforms the baselines and existing works in terms of performance, accuracy, and resource utilization in one aspect or another. Our HTS achieves up to 42.64% computational savings (GFLOPs) in ViT-base with an accuracy loss of only 1.45% on ImageNet compared to the baseline (no pruning).

VIII. Acknowledgment

This work was supported by the SERB, Government of India, under Project SRG/2023/002108.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[2] P. Dong, M. Sun, A. Lu, Y. Xie, K. Liu, Z. Kong, X. Meng, Z. Li, X. Lin, Z. Fang *et al.*, “HeatViT: Hardware-Efficient Adaptive Token Pruning for Vision Transformers,” in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 442–455.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE,” *arXiv preprint arXiv:2010.11929*, 2020.

[4] 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 *European Conference on Computer Vision*. Springer, 2022, pp. 396–414.

[5] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang *et al.*, “SPViT: Enabling Faster Vision Transformers via Soft Token Pruning,” in *European conference on computer vision*. Springer, 2022, pp. 620–640.

[6] S. Lee, K. Cho, E. Kwon, S. Park, S. Kim, and S. Kang, “ViT-ToGo: Vision Transformer Accelerator with Grouped Token Pruning,” in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–6.

[7] Z. Li, M. Sun, A. Lu, H. Ma, G. Yuan, Y. Xie, H. Tang, Y. Li, M. Leeser, Z. Wang, X. Lin, and Z. Fang, “Auto-ViT-Acc: An FPGA-Aware Automatic Acceleration Framework for Vision Transformer with Mixed-Scheme Quantization,” in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 109–116.

[8] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie, “Not All Patches are What You Need: Expediting Vision Transformers via Token Reorganizations,” *arXiv preprint arXiv:2202.07800*, 2022.

[9] S. Long, Z. Zhao, J. Pi, S. Wang, and J. Wang, “Beyond Attentive Tokens: Incorporating Token Importance and Diversity for Efficient Vision Transformers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 334–10 343.

[10] S. Nag, G. Datta, S. Kundu, N. Chandrachoodan, and P. A. Beerel, “ViTA: A Vision Transformer Inference Accelerator for Edge Applications,” *arXiv preprint arXiv:2302.09108*, 2023.

[11] B. Pan, R. Panda, Y. Jiang, Z. Wang, R. Feris, and A. Oliva, “IA-RED^2: Interpretability-Aware Redundancy Reduction for Vision Transformers,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 898–24 911, 2021.

[12] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, “DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification,” *Advances in neural information processing systems*, vol. 34, pp. 13 937–13 949, 2021.

[13] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 430–443.

[14] M. Sun, H. Ma, G. Kang, Y. Jiang, T. Chen, X. Ma, Z. Wang, and Y. Wang, “VAQF: Fully Automatic Software-Hardware Co-Design Framework for Low-Bit Vision Transformer,” *arXiv preprint arXiv:2201.06618*, 2022.

[15] T. Wang, L. Gong, C. Wang, Y. Yang, Y. Gao, X. Zhou, and H. Chen, “ViA: A Novel Vision-Transformer Accelerator Based on FPGA,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4088–4099, 2022.

[16] X. Xu, S. Wang, Y. Chen, Y. Zheng, Z. Wei, and J. Liu, “GTP-ViT: Efficient Vision Transformers via Graph-based Token Propagation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 86–95.

[17] 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*, 2022, pp. 10 809–10 818.