TA-ASF: Attention-sensitive Token Sampling and Fusing for Visual Transformer Models on the Edge

Junquan Chen*†‡, Xingzhou Zhang‡§✉, Wei Zhou*†, Weisong Shi¶
*Engineering Research Center of Cyberspace, Yunnan University, Kunming 650091, China
†School of Software, Yunnan University, Kunming 650091, China
‡Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
§University of Chinese Academy of Sciences, Beijing 100190, China
¶Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA
chenjunquan@mail.ynu.edu.cn, zhangxingzhou@ict.ac.cn, zwei@ynu.edu.cn, weisong@udel.edu

***Abstract*—Vision Transformers (ViTs) have made significant progress in achieving performance comparable to traditional convolutional neural networks in computer vision tasks. However, high computational complexity restricts their application to resource-constrained edge devices. Previous methods for pruning redundant tokens have shown that it is possible to balance performance and computational cost by reducing the number of tokens. Unfortunately, simply removing redundant tokens often leads to the loss of crucial information. To address this issue, we propose a novel token compression scheme called TA-ASF. This scheme considers both the global role of low-importance tokens and the redundancy among similar tokens. TA-ASF employs novel approaches for token sampling and fusion, which are directly applicable to ViTs without introducing additional trainable parameters. A comprehensive evaluation against several edge devices demonstrates our method effectively reduces model complexity while preserving Top-1 accuracy. Experimental results show that on the ImageNet dataset, the proposed method reduces FLOPs by 37% and increases throughput by 1.48 times on the DeiT-S model, with only a 0.1% decrease in accuracy. Specifically, on the DeiT-B model, the proposed method decreases FLOPs by 35% and increases throughput by 1.52 times while maintaining the same accuracy.**

***Index Terms*—Model Compression, Vision Transformers, Edge Computing.**

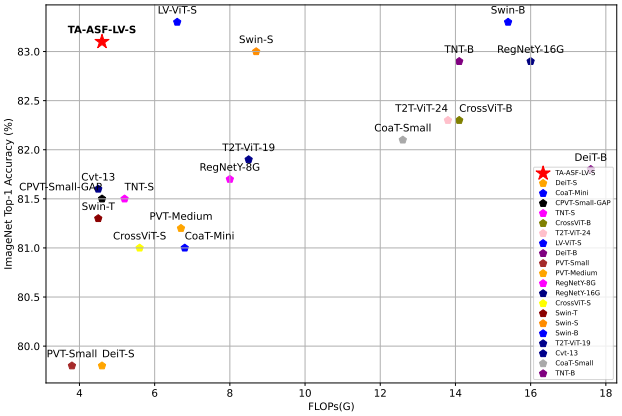

Fig. 1. Performance of accuracy-flops for different vision transformer models on the ImageNet dataset. TA-ASF enabled vision model, TA-ASF-LV-S in the upper left corner, maintains high accuracy while reducing the computing demand(FLOPs).

I. Introduction

As one of the basic structures of the large foundation models, Vision Transformers (ViTs) [1] has significantly changed the landscape of computer vision, utilizing self-attention mechanisms [2] to process image data and achieve state-of-the-art performance on a variety of tasks. ViTs achieve significant performance in tasks such as image classification [3], [4], [5], object detection [6], [7], [8] and semantic segmentation [9], [10], [11] by partitioning images into sequential patches similar to those used in Natural Language Processing (NLP) [12]. Despite their superior performance, the computational and memory demands of ViTs pose significant challenges for deployment on edge devices.

Research indicates that model lightweighting is crucia for edge devices [13], [14], [15], [16], [17]. Although ViTs offer remarkable advancements, the computational and memory demands of their core self-attention mechanism [2] grow quadratically with the number of patches. This poses significant challenges for deployment on platforms with limited computational power and memory capacity. For example, a typical ViT model, such as the Swin-L [11] model, contains approximately 197 million parameters and requires about 104 GFLOPs for a single forward propagation when processing a 384x384 resolution image. In contrast, the NVIDIA Jetson Nano, a common edge computing device, has a theoretical computational power of only 472 GFLOPs, allowing it to process up to about four image frames per second under ideal conditions. However, due to practical factors such as system overhead, memory bandwidth limitations, and computational efficiency, the actual processing speed is often much lower than the theoretical value. This mismatch of resources highlights the need to develop more efficient ViT models and optimization algorithms.

However, there are several key challenges that we have to consider when increasing the speed of the model. **(1) Trade-off between throughput and accuracy.** Accelerating a model usually means increasing the throughput of the model, but striking a balance between model speed-up and performance

maintenance is an extremely complex challenge. Simply pursuing acceleration is likely to result in a degradation of model performance, especially in terms of accuracy. Common methods to model acceleration include pruning [18], [19] and quantization [20], [21]. Pruning reduces computational requirements by removing less important elements, potentially weakening the model's expressiveness and degrading performance on complex tasks. Quantization reduces model weights and activation values from 32-bit floating-point numbers to lower bit-widths, such as 8-bit integers, significantly reducing computational and storage needs. However, quantization errors can degrade overall model performance, particularly in high-precision tasks. Different techniques have their own advantages and disadvantages, how to use the right technique to adapt the balance between model acceleration and performance maintenance in ensuring the efficiency of the model while maximizing the maintenance of its performance is a challenge.

Another key challenge is **(2) how to design a generalized method for more models.** Many current techniques and optimization methods are tailored for specific models or tasks, limiting their applicability. For example, MobileViT [22] effectively reduces the computational cost of a model and improves inference speed on mobile devices by combining convolution and self-attention mechanisms, but its specific optimization methods may not be easy to directly migrate to other models. This highlights the importance of designing a generalized acceleration method. Especially in real-world applications, the development of a general and efficient acceleration method is particularly important due to the varying model requirements of different tasks and scenarios.

To solve the above two problems, one of the most straightforward ways is to reduce the computational complexity of the transformer structure. [4], [11], [23], [24], [25] improved the transformer structure and reduced the computational complexity of the model through the effective design of the attention mechanism and the feed-forward network. For example, EfficientViT [24] introduces a multiscale linear attention module that reduces computational complexity from quadratic to linear. Unlike the conventional use of softmax attention in ViT [1], it employs lightweight ReLU linear attention for global visual field perception. However, this method requires changes to specific structures and lacks generalization.

At the same time, there is a strong correlation between the complexity of the model and the number of tokens in ViT, thus reducing the computational complexity, another method is to reduce the number of tokens. [26], [26], [27], [28], [29] evaluate the importance of tokens by designing different token scoring strategies or by adding a score prediction module to remove unimportant tokens based on their scores. For example, [26] chooses to evaluate the importance scores between classes of attention to reorganize the image tokens. Unlike these works, Tome [30] chooses to evaluate the recognition between tokens and merge the tokens with high recognition together evenly.

However, the methods mentioned above either require the addition of extra modules, leading to increased costs, or they address only quantitative or similarity redundancy, which is not a comprehensive solution. Therefore, inspired by these works, this paper proposes **TA-ASF**, a novel compression method with high **T**hroughput and **A**ccuracy, which consists of two components: **A**ttention-sensitive token **S**ampling and token **F**using. TA-ASF effectively addresses both types of redundancy in ViT models without introducing any new trainable modules. As illustrated in Figure 2, we make no modifications to the handling of input token sequences in standard Transformer layers. However, upon entering specific layers, we sort the input tokens based on attention weights. Rather than simply discarding unnoticed tokens, we sample them according to their attention scores. Subsequently, we design a specialized token fusion technique to process the sampled tokens based on their similarity metrics. Importantly, our method does not introduce additional trainable parameters, allowing for seamless integration into existing pre-trained ViT models. We extensively evaluated our method on representative ViT models such as DeiT [4] and LV-ViT [31] using the ImageNet dataset [32]. As depicted in Figure 1, when TA-ASF is integrated into LV-VIT-S, we achieve competitive results among many state-of-the-art models, demonstrating a balanced trade-off between accuracy and computational complexity. By effectively addressing the number and similarity redundancies while avoiding the complexity inherent in traditional compression methods, our method significantly reduces the computational resource requirements with minimal impact on model performance. This provides a promising solution for deploying ViT models in resource-constrained environments.

Our main contributions are summarized as follows:

- This paper proposes a more **thorough lightweighting** method for edge vision transformer models while maintaining high accuracy, TA-ASF. Compared with the current methods, which can only remove one redundant, TA-ASF removes both quantitative redundant and similarity redundant information.
- TA-ASF presents **generality** and can be applied to multiple models. It is optimized for the image input layer, making it suitable for a class of vision transformer models, rather than specifically for a single model. TA-ASF enabled LV-ViT-S and DeiT-S models reduced the computational command (FLOPs) by 30.3% and 37%, respectively, while maintaining stable accuracy.
- TA-ASF is **adaptable** to edge devices, which is validated on multiple edge devices, including Edge Server with V100 GPU, NVIDIA Jetson Nano, Jetson Orin NX, and Jetson AGX Orin. Taking TA-ASF-DeiT-B running on Edge Server and Jetsion Orin NX as examples, the throughput is increased by 1.52 and 1.42 times, respectively.

II. Preliminaries

A. Patches embedding

TA-ASF method is applied primarily directly to standard ViTs [1] and its variants. Therefore, we first provide a com-

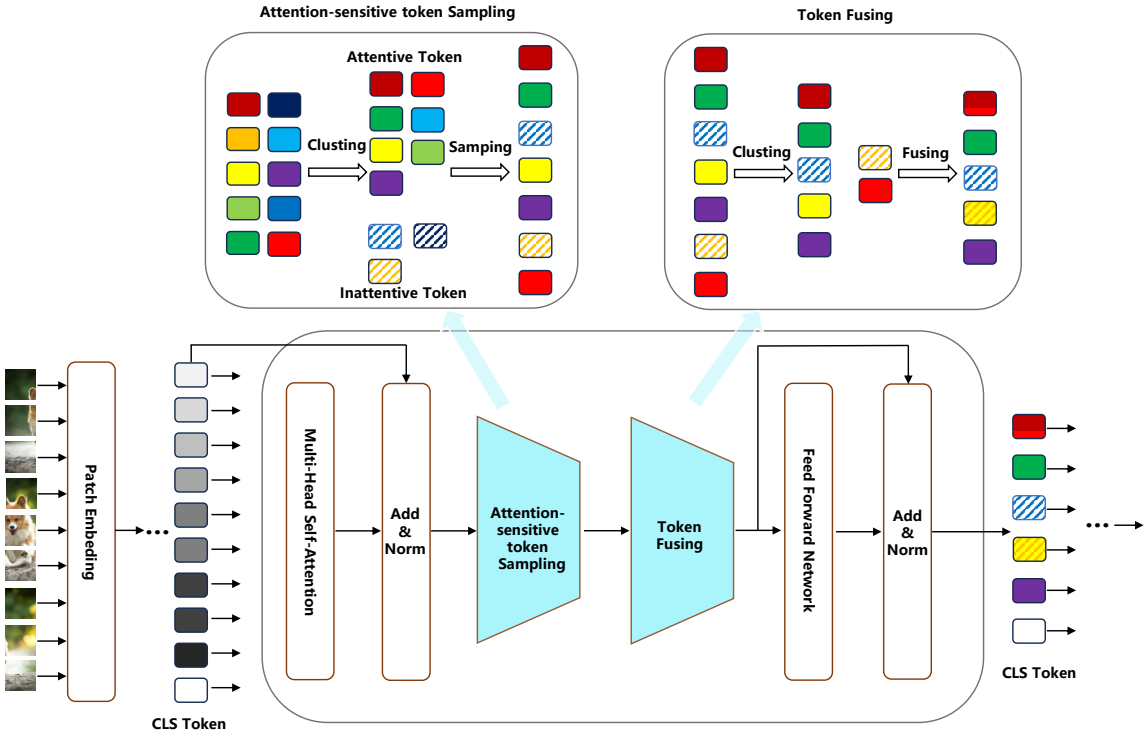

Fig. 2. The general framework of the proposed method is illustrated within a single transformer encoder. For the input sequence after the Multi-Head Self-Attention (MHSA) operation, we use the Attention-sensitive token Sampling module to compute the scores of the tokens based on class markers. According to these scores, the tokens are categorized into attentional and inattentional groups. Within each group, the tokens are sampled and combined into a new token group, which is then inputted into the fusing module. In the fusing module, we calculate the similarity between the tokens. Based on this similarity, tokens are fused through one-to-one matching to produce the final output, which is then input to the next transformer encoder.

prehensive overview of ViTs, followed by a discussion in the next section on how our method is integrated into ViTs.

The overall framework of the ViT model is illustrated in Figure 3. Initially, Transformers are applied in the NLP domain, where the input consists of 1D word embeddings. In the vision domain, the input changes from 1D word embeddings to 2D images. To handle 2D images, during the patch embedding stage, the image $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$ is reshaped into a series of flattened 2D patches $\mathbf{X}_p \in \mathbb{R}^{N\times\left(P^2C\right)}$. Here, (H, W) represents the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the number of generated patches, which is also the effective input sequence length for the Transformer. Each patch is projected into a token embedding using a trainable linear layer for tokenization. An additional class token $\mathrm{x}_{\mathrm{cls}}$ is added to the set of image tokens, which is responsible for aggregating global image information and performing the final classification. A learnable positional embedding vector E_{pos} is also added to retain positional information. The final input can be represented as:

$$\mathrm{X}_p = [\mathrm{x}_{\mathrm{cls}}; \mathrm{x}_1; \ldots; \mathrm{x}_{\mathrm{N}}] + \mathrm{E}_{\mathrm{pos}} \tag{1}$$

where x_i represents the $i-th$ token. Finally, X_p is fed into the sequentially stacked Transformer encoder, which consists of a multi-head self-attention (MHSA) layer and a feed-forward network (FFN).

B. MHSA And FNN

In the MHSA, the input tokens are linearly mapped and further packed into three matrices: query (Q), key (K), and value (V). The attention operation is defined as follows:

$$Attention = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)\mathrm{V} \tag{2}$$

where d represents the feature-length per single head. The result of Softmax $(QK^T/\sqrt{d})$ is a matrix known as the attention map. The first row of the attention map indicates the attention from $\mathrm{x}_{\mathrm{cls}}$ to all tokens, which is used to determine the importance of each token. The outputs of H such attention maps are concatenated to form the final output of the Multi-Head Self-Attention (MHSA). Subsequently, the output of MHSA is fed into the FFN, which consists of two fully connected layers with an intermediate GELU [33] activation layer. The MHSA and FFN together constitute a Transformer encoder, and a ViT model is constructed by stacking multiple such encoders sequentially. In the final Transformer encoder of the model, the $\mathrm{x}_{\mathrm{cls}}$ token is extracted to predict object categories based on its values. Further details on Transformers can be found in [2].

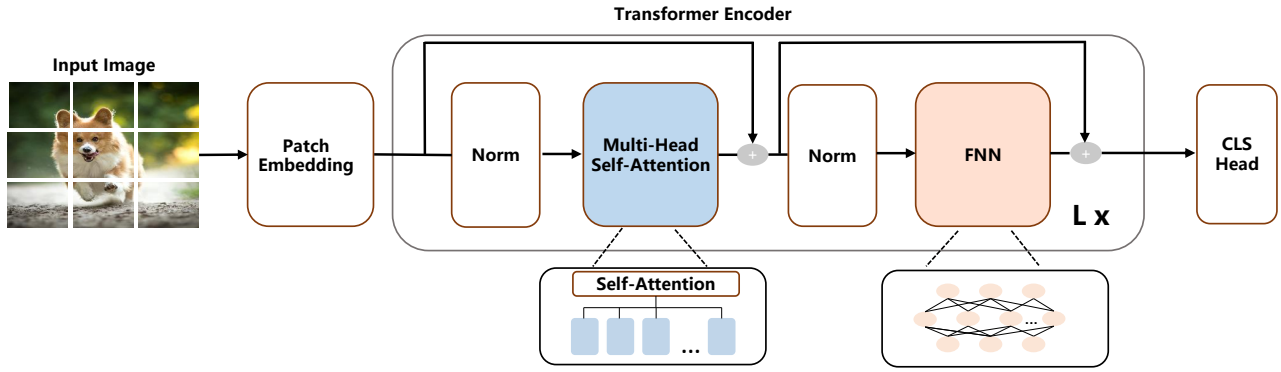

Fig. 3. The structure diagram of the standard ViT model, which consists of a patch embedding layer, several transform encoders, and a classification header. Each transform encoder contains one MHSA and FNN operation. The model size mainly depends on the number of encoders, embedding dimensions, number of attention headers, and FNN dimensions.

C. Computational complexity

The dimension of the input token sequence is $N \times C$, where N is the number of tokens, and C is the embedding dimension of each token. The computational cost of ViT is primarily concentrated in the Multi-Head Self-Attention (MHSA) and Feed-Forward Neural Network (FFN) modules. Specifically, the costs are $O(4NC^2 + 2N^2C)$ and $O(8NC^2)$, respectively. The term $O(4NC^2)$ arises from projection computations, which include linear transformations from input to query (Q), key (K), and value (V) vectors. The $O(2N^2C)$ term reflects the complexity of computing the self-attention mechanism, as each token must compute attention weights with every other token. The $O(8NC^2)$ term corresponds to the linear transformations within the two linear layers of the FFN. Given that $2N^2C$ scales quadratically with N, reducing N can significantly lower this portion of the cost. However, reducing C diminishes the feature representation capacity of each token, potentially causing a decline in model performance. Conversely, reducing N has minimal impact on model performance [1], [4]. Therefore, compressing N is a natural consideration.

III. Method

A. Overview

Unlike existing methods that focus solely on deleting or merging tokens, our method takes into account both the importance of attention within the image and the similarity between tokens. This comprehensive analysis of the input token sequence enables effective token compression. Our method achieves an optimal balance between accuracy and FLOPs in the ViTs model.

As illustrated in Figure 2, our method can be seamlessly integrated into any transformer encoder within the model without necessitating specific design modifications to the original architecture. Furthermore, it does not involve any trainable parameters, making it applicable in both training and non-training scenarios. Remarkably, it delivers impressive results even without customization or fine-tuning. In this section, we first present the token pruning and token fusing techniques employed in our method, followed by a detailed description of our overall process.

B. Attention-sensitive token Sampling

To effectively introduce our token pruning method, we will start with a brief overview of the basic token pruning process. Typically, token pruning involves two main steps: token scoring and token selection. In the token scoring step, each token is assigned a score reflecting its importance to the task at hand. The token selection step uses these scores to decide which tokens to retain and which to discard.

Token scoring: In prior research, several scholars have proposed non-parametric scoring strategies [26], [31]. Inspired by [26], we recognized that the result of $\mathrm{Softmax}\left(\mathcal{Q}\mathcal{K}^T/\sqrt{d}\right)$ in the attention mechanism effectively represents token importance. Consequently, we selected ATS [34] as our scoring strategy because it simultaneously accounts for the value matrix V and the attention matrix A, making it a more comprehensive token scoring metric.

$$\mathrm{A} = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \tag{3}$$

$$S = \frac{A_{1,j} \times \|V_j\|}{\sum_{i=2} A_{1,i} \times \|V_i\|} \tag{4}$$

Where $i, j \in (2 \ldots N)$, S is the final score matrix, $A_{1,j}$ represents the importance of input token j to the output classification token, S_j denotes the significance score of the $j-th$ token, and V_j represents the value of the $j-th$ token. As the classification token $A_{1,1}$ must be retained and is not evaluated, the indexing starts from 2 rather than 1.

Token selection: Traditional token selection methods typically employ a Top-K selection strategy. This involves sorting all tokens, except for the classification token, by their scores and selecting the top K tokens. All tokens ranked beyond K are discarded. Inspired by [28], we recognize that token importance can change across different stages, a low score at one stage only indicates weak local importance and does not reflect global importance. Considering that tokens of varying

importance levels contribute differently to the final task, we propose a novel selection algorithm, Algorithm 1, which employs the concept of quota sampling to select a specific number of tokens from the set originally intended for exclusion. By sampling from the low-scoring set, the risk of discarding tokens that may later become important is minimized, while more high-scoring tokens are retained. Assume K represents the number of tokens we aim to retain. After sorting the token set M based on their importance scores, it is divided into two subsets, M_h and M_l. Tokens are then selected from each subset in proportion to their respective sizes relative to the total set size, ensuring that the total number of tokens selected from both subsets remains K. Next, set a starting insertion position p, and calculate the arithmetic sequence's common difference q based on the quantities N_r and N_d extracted from subsets M_h and M_l, respectively, following equation (5)

$$q = \frac{(N_r - p)}{N_d} \tag{5}$$

To ensure that low-scoring tokens are evenly distributed in the high-scoring set, we usually choose p to be at the initial position of the high-scoring set. Finally, based on the q and p insert tokens extracted from M_l into M_h to form the new set M_n, which will be used in the next step.

Algorithm 1 Algorithm Description of Attention-sensitive token Sampling

Input token set: M, keep rate: r, first insertion position p, score:S

_, index = sort(S) ▷ Sort the scores in descending order to get the index of the sorted scores
$L = len(M)$ ▷ Get the number of tokens in a set
$L_r = L * r$
$L_d = L * (1 - r)$
$index\_r = index[0, 1...L_r], index\_d = index[L_r...L]$ ▷ Get the index of the target token
$M_h = M[index\_r]$
$M_l = M[index\_d]$ ▷ Get the partitioned set of tokens according to their respective indexes
$N_r = \frac{L_r}{L} * L_r$
$N_d = \frac{L_d}{L} * L_r$ ▷ Calculate the number of samples in the respective set
Take out the corresponding number of tokens in the respective sets based on N_r and N_d to get M'_h andM'_l
$q \leftarrow N_r, N_d, p$ based on Equation(5)
for x in M'_l **do**
    $M'_h \xleftarrow{insert} x$ base on q and p ▷ Insert a new token at the corresponding position of M'_h
end for
Get new M_n =M'_h

Output The new set of processed tokens M_n

C. Token Fusing

After obtaining the processed token set M_n we further compress the tokens by dividing M_n into subsets M_r and M_d according to a proportion p. Since the position of tokens in M_n inherently reflects their scores, tokens in M_r are considered more important. To preserve more important tokens and reduce the overall token count, we opt to merge tokens from M_d into M_r without generating additional tokens as done in [35] [26]. We introduce Algorithm 2 to match and merge tokens from these two subsets.

For the given sets M_r and M_d, K_r and K_d denote the indices of tokens in M_r and M_d, respectively. Using a similarity matrix $C_{i,j}$ record the similarity values between tokens in M_d and M_r, where $i \in K_d$ and $j \in K_r$. For any token x_i in M_d, based on the values in $C_{i,j}$ from M_d to M_r, we identify the most similar token x_j in M_r. Subsequently, we perform pairwise one-to-one merging between them. As for obtaining the similarity matrix, we compute it using low computational complexity cosine similarity. The equation for computing the similarity matrix is:

$$C_{i,j} = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, \text{ for } i \in K_d, j \in K_r \tag{6}$$

where x_i and x_j represent tokens from different sets. Each token in M_d can identify the token with the highest similarity from M_r based on the values in C, which serves as the basis for subsequent fusion. The elements of the similarity matrix are derived directly from input features, ensuring no additional parameters are introduced. Moreover, due to the low computational complexity of cosine similarity, the additional computational load for calculating similarity between tokens is negligible during the inference process of the entire model.

When merging tokens, differences between different tokens can lead to feature dispersion and potentially affect model performance if simply averaged. Inspired by the reweighted token aggregation based on token importance scores in [26], we perform weighted aggregation based on token similarity. Taking into account the importance scores of tokens, we

Algorithm 2 Algorithm Description of Token Fusing

Input token set: M_n, keep rate: r

$L = \text{len}(M_n)$ ▷ Get the number of tokens in a set
$L_r = L * r$
$L_d = L * (1 - r)$
$M_r, M_d \leftarrow \text{divide}(M_n)$ ▷ Division based on length L_r and L_d
K_r, K_d representing the indexes of M_r and M_d
Calculate the recognition matrix C based on Equation (6)
$\text{index}_{\max} = \arg\max(C)$ ▷ Find the index with the highest similarity
for x in M_d **do**
    Calculate w_x based on Equation (8) ▷ Calculate the weight of x based on the similarity value
    Update y based on Equation (7) ▷ Fuse token x to the corresponding similar token according to w_x
end for
Get new M_r

Output The new set of processed tokens M_r

prioritize x_j from M_r as the primary token and x_i from M_d as the secondary token, assigning weight w_i to x_i based on their similarity. The merging is performed according to the equation (7), where y_j represents the token updated after merging.

$$y_j = x_j + w_i x_i \tag{7}$$

The weight w_i is computed based on equation (8), primarily determined by the similarity matrix C.

$$w_i = \frac{\exp(C_{i,j^*})}{\sum_{j=1}^{n} \exp\left(C_{i,j}\right)} \tag{8}$$

Where j^* represents the index of the token with the highest similarity, and n denotes the number of tokens in the set M_r.

Through the above method, we have updated the token set M_r, while tokens that have not undergone merging remain unchanged. Our compression significantly reduces the number of tokens compared to the original set M, with minimal impact on model performance. Subsequent experimental results validate the superiority of our method.

D. Whole Comoressing Process

For a given Vision Transformer (ViT) model, our compression process is outlined in Algorithm 3. Initially, based on predefined hyperparameters in the $p-list$, specific positions within the model are identified for compression. During the model's inference process, when these designated positions, referred to as lottery layers, are reached, the input data is compressed using the procedures described in Algorithms 1 and 2. The compressed input is then forwarded to the subsequent Transformer encoder. With each lottery layer, the number of tokens is reduced, contributing to a decrease in computational complexity and memory usage. In contrast, during regular layers, the model processes the input tokens sequentially without modifying their count, thereby maintaining a consistent number of tokens at this stage. Throughout the entire process, token compression operations are exclusively conducted at lottery layers, ensuring that the efficiency gains from compression are maximized while preserving the integrity and performance of the model.

Algorithm 3 Algorithm Description of TA-ASF

Input data: M, keep rate: $r1$, keep rate: $r2$, Model: $model$, com_list: p_list, first insertion position: p

1: **for** each i in $model.blocks$ **do**
2: **if** i is in p_list **then**
3: Mn is obtained by executing Algorithm 1
4: Mr is obtained by executing Algorithm 2
5: **else**
6: Performing the operations defined in the original model yields the output M
7: **end if**
8: **end for**
9: Extract the classification markers x_{cls}

Output The final classification result x

IV. Experiment

A. Data and evaluation metrics

Dataset: All experiments are conducted using the ImageNet-1K dataset (ILSVRC2012) [32]. This dataset comprises approximately 1.28 million training images and 60,000 test images, spanning a total of 1,000 categories. The object categories encompass a diverse range, including animals, plants, everyday objects, and various types of equipment. Additionally, the distribution of images across these categories is relatively balanced.

Metrics: We use official tools and customized code to measure the following four key metrics to evaluate TA-ASF:

- Top-1 Acc: Top-1 accuracy refers to the scenario in which only the predicted category with the highest probability is considered in the model's output. If this predicted category matches the true category, the prediction is deemed correct, thereby contributing to the model's accuracy. Top-1 accuracy is calculated by dividing the number of correct predictions by the total number of samples in the test set.
- FLOPs: The number of floating-point operations (FLOPs) required by the model to process an image serves as a metric for assessing the computational resource consumption of the model. We utilize the $profile\_macs$ tool from the Python library $torchprofile$ to calculate the FLOPs value. This is done by inputting a standard 224x224 size image into the model.
- Throughput: The number of images processed by the model per unit of time serves as an indicator of the model's running speed. To measure throughput, we record the total time taken to process a fixed number of images and then calculate the ratio of the number of images to the total processing time.
- Params: The total number of trainable parameters in the model. To calculate this, we traverse the model and count the parameters in each layer, summing these values to obtain the overall parameter count.

B. Standard Benchmarks for Comparison

In this section, we provide a brief overview of the standard benchmark models used in our experiments. We divide these models into two broad categories: models that focus on optimizing model input data and models that aim to improve the model architecture itself.

Optimization of input data to the model:

- DynamicViT [27] employs a lightweight prediction module to predict importance scores for tokens. Tokens are then selected based on these predicted scores.
- Evo-ViT [28] addresses the spatial structure changes during the pruning process by updating tokens using two distinct methods: fast and slow updates.
- SPViT [35] proposes an attention-based dynamic selector for multiple tokens and introduces a soft pruning technique to merge tokens with less information.

TABLE I
Configuration details for four different devices.

	Edge Server	Jetson AGX Orin	Jetson Orin NX	Jetson Nano
CPU	Intel® Xeon® Gold 6138	Arm® Cortex®-A78AE v8.2	Arm® Cortex®-A78AE v8.2	ARM® Cortex®-A57
CPU Max Frequency	3.7GHz	2.2GHz	2.0GHz	1.43GHz
GPU	5120-core TESLA V100 NVIDIA Volta@1.38GHz	2048-core NVIDIA Ampere@1.3GHz	1024-core NVIDIA Ampere@918MHz	128-core NVIDIA Maxwell™ @921MHz
RAM	32GB	64GB	16GB	4GB
Power	250W	15W- 60W	10W- 25W	5W - 10W
AI Performance	14TFLOPS	275TOPS	100TOPS	472GFLOPS
Operating System	Ubuntu 16.04.7	Ubuntu 20.04.6	Ubuntu 20.04.6	Ubuntu 18.04.6
Deep Learning Framework	PyTorch 1.12.1	PyTorch 1.13.0	PyTorch 1.13.0	Pytorch 1.8.0
CUDA	11.2	11.4	11.4	10.4

- SViTE [36] obtains a sparse network through dynamic extraction and training of a sparse subnetwork for model inference.
- EViT [26] reorganizes image tokens during the model's feed-forward process, merging inattentive tokens into a new token.
- IA-RED^2 [29] introduces an interpretable token selection module that systematically identifies crucial tokens.
- Tome [30] proposes a strategy for merging tokens based on their similarities.
- ATS [34] introduces an efficient token-scoring method that relies solely on the model's original parameters.
- PS-ViT [37] implements a top-down token selection method that discards tokens starting from the output layer.

Improvement of models

- DeiT [4] introduces a distillation token that guides self-model training with guidance from a teacher model.
- PVT [38] specifically designs a convolution-free backbone network for image classification.
- RegNetY [39] introduces a new network design paradigm and develops an efficient network structure.
- CrossViT [40] proposes a dual-branch transformer that integrates patches of different sizes for inference.
- Swin [11] introduces a hierarchical transformer using shifted windows for computation.
- T2T-ViT [5] proposes a transformer that recursively aggregates adjacent tokens into one token.
- Cvt [41] introduces an efficient transformer incorporating convolution operations.
- CoaT [42] proposes a transformer with cross-layer learning capabilities.
- TNT [43] introduces a new structure embedding transformers within transformers.
- LV-ViT [31] adds a new trainable token to the transformer, converting the image classification problem into a token recognition problem.
- CPVT [44] adds a conditional positional encoding structure to the transformer.

C. Experimental setup

As shown in Figure 4, we conducted tests on four different edge platforms. We use an edge server equipped with Tesla V100 to represent a high-performance platform, while the Jetson AGX Orin, Jetson Orin NX and Jetson Nano, are used to represent resource-constrained edge devices. The detailed configurations of these devices are shown in Table I.

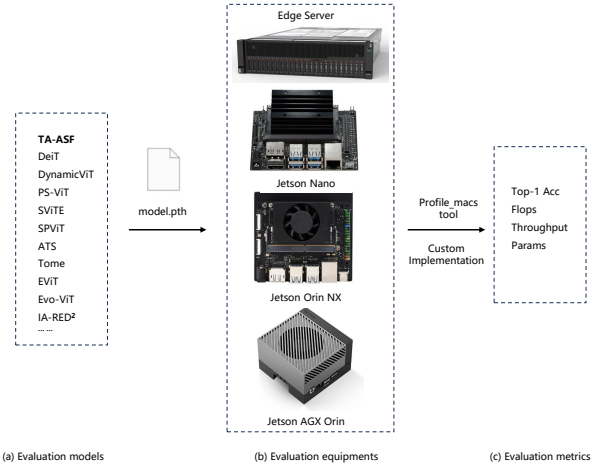

Fig. 4. Four different experimental devices for deploying TA-ASF in the experiment

D. Implementation details

We conducted experiments on standard ViTs [4] (including DeiT-T, DeiT-S, and DeiT-B) and the ViT variant LV-ViT-S [31]. Following [26], we employ our method to the 4^{th}, 7^{th}, and 10^{th} layers of the DeiT-T/S/B models and the 4^{th}, 6^{th}, and 14^{th} layers of the LV-ViT-S model. For the comparative experiments on DeiT-T/S/B and LV-ViT-S, we report the performance of our method in both off-the-shelf and fine-tuned scenarios. Following the method of [27], we initialize the backbone models with the officially pre-trained ViT weights. Regarding training strategies and optimization methods, we adhere to the settings described in the original DeiT and LV-ViT papers, as our method does not modify the parameters, making the original strategies highly compatible. All images used for training and testing have a resolution of 224 $\times$ 224. All experiments were conducted on NVIDIA GPUs using the PyTorch deep learning framework. Model throughput, unless

otherwise specified, was measured on a single NVIDIA V100 GPU with a fixed batch size of 256.

E. Main results

1) Comparisons with existing methods: We compared our method with existing methods. Despite not adding additional parameters like [27], or applying complex token reorganization techniques like [26], our method achieves better results with comparable or lower computational costs, establishing a new state-of-the-art performance. As shown in Table II, we tested the Top-1 accuracy, Params, FLOPs, and Throughput of each model. From the table, we can see that for the standard model DeiT-S, our method only reduces the top-1 accuracy by 0.1% while decreasing computational cost by 37%. Notably, for the standard model DeiT-B, our method maintains the top-1 accuracy while reducing the computational cost by 35% and increasing the throughput by 1.52 times. Our method also performs well when applied directly to models without fine-tuning.

2) Accuracy at different computational volumes: We applied our method to DeiT-S and compared it with existing methods under different computational budgets. As shown in Figure 5, our method consistently achieves the best performance across various computational levels. When the model's FLOPs are reduced to 2.3G, which is only half of the baseline's computational load, our method achieves a classification accuracy of 79.0%, close to the original model's accuracy. In contrast, the classification accuracies of other methods are around 78%. When the computational load increases to 2.6G, our method's accuracy reaches 79.5%, only 0.3% lower than the baseline while reducing the computational load by 43.5%. At this point, other methods still have accuracies below 79%. These results clearly show that by simultaneously considering two types of redundancy compression, we can reduce the number of tokens while retaining more of the original valid information, thereby minimizing the negative impact on model performance.

3) Deployment on Edge Devices: To further evaluate the performance of our method on resource-constrained edge devices, we conducted experiments on Jetson Nano, Jetson Orin NX and Jetson AGX Orin. As shown in Table III, our method accelerates the standard models DeiT-T/S/B to varying degrees. On Orin NX and Orin AGX Orin, the speeds of DeiT-T/S are increased by 1.35 and 1.46 times, respectively, compared to the baseline. For DeiT-B, the speeds are increased by 1.41 and 1.43 times, respectively. Notably, on Orin NX, we increased the running speed of the large model DeiT-B from the baseline of 36 img/s to 51 img/s, meeting real-time inference requirements. Although the numerical increase may seem modest, it allows us to choose high-complexity large models for scenarios requiring high-precision real-time inference, rather than smaller models that sacrifice accuracy. Due to computational constraints on the Jetson Nano, we deployed our method on the DeiT-T model with a batch size of 32. We observed a notable increase in inference speed from 30 img/s to 42 img/s, achieving a 1.4x improvement. Our method makes fast inference possible on edge devices. Furthermore, we observe that when the computational performance of edge devices is similar, the inference speed of the model is influenced not only by ai performance but also by other factors such as power consumption and CPU performance. However, when there is a significant difference in computational performance, it becomes the predominant factor affecting inference speed.

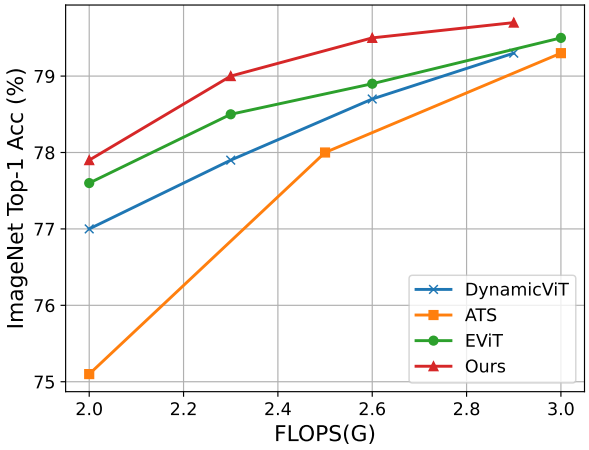

Fig. 5. Performance of the DynamicViT, ATS, and EViT methods, as well as our method, when applied to the DeiT-S model at different FLOPs.

4) Application on a variant of ViTs: In addition to the standard ViT, numerous researchers have made significant improvements to ViT's performance by altering the initial architecture or optimizing strategies. To further demonstrate the plug-and-play nature and superior performance potential of our method, we applied it to LV-ViT-S and conducted comparative experiments with these methods. As shown in Table IV and Figure 1, our TA-ASF-LV-S achieves better performance in terms of accuracy-complexity trade-offs compared to many state-of-the-art vision transformers. Our method achieves higher classification accuracy with lower computational cost and without increasing the number of parameters, further proving the superiority of our method over other token compression methods.

F. Ablation experiment

1) Effectiveness of each module: As shown in Table V, we demonstrate the effectiveness of each module by individually adding different sub-modules under varying computational budgets. i) Token Selection Module Only: During the pruning stage, tokens with low attention scores are discarded based on a top-k selection method, serving as the baseline. ii) Sampling Module Only: During the pruning stage, tokens are divided into two sets based on importance scores, and a new set is formed by sampling from these sets proportionally. iii) Fusion Module Only: Tokens are matched based on similarity and fused into new tokens with certain weights. iv) Both Sampling and Fusion Modules: During the pruning stage, tokens are discarded based on the sampling method to form a new token

TABLE II
Comparison with existing state-of-the-art DeiT token pruning methods on ImageNet. The markers indicate results directly applied to the DeiT model without fine-tuning (off-the-shelf). We primarily compare four metrics: Top-1 accuracy, FLOPs, Parameters, and Throughput. 'Top-1 Acc ↓' denotes the reduction ratio of Top-1 Acc, and 'Throughput ↑' denotes the increase ratio of Throughput.

Model	Method	Params (M)	FLOPs (G)	Top-1 Acc (%)	Top-1 Acc ↓ (%)	Throughput (img/s)	Throughput ↑ (%)
DeiT-T	Baseline [4]	5.6	1.3	72.2	0.0	2700	0.0
	DynamicViT [27]	5.9	0.9	71.2	1.0	3603	33.4
	Evo-ViT [28]	5.6	0.8	72.0	0.2	3654	35.3
	PS-ViT [37]	5.6	0.7	72.0	0.2	3576	32.4
	SViTE [36]	4.2	0.9	70.1	2.1	2836	5.0
	SPViT [35]	5.6	1.0	72.2	0.0	-	-
	EViT [26]	5.6	0.8	71.9	0.2	3587	32.9
	TA-ASF(off-the-shelf)	**5.6**	**0.8**	**70.0**	**2.2**	**3650**	**35.2**
	TA-ASF	**5.6**	**0.8**	**72.1**	**0.1**	**3650**	**35.2**
DeiT-S	Baseline [4]	22.1	4.6	79.8	0.0	990	0.0
	DynamicViT [27]	22.8	3.0	79.3	0.5	1430	44.4
	IA-RED^2 [29]	22.1	3.2	79.1	0.7	1362	37.6
	Evo-ViT [28]	22.1	3.0	79.4	0.4	1431	44.5
	PS-ViT [37]	22.1	2.6	79.4	0.4	1356	37.0
	EViT [26]	22.1	3.0	79.5	0.4	1416	43.0
	Tome [30]	22.1	2.7	79.4	0.3	1552	56.8
	ATS [34]	22.1	2.9	79.7	0.1	1382	39.6
	TA-ASF(off-the-shelf)	**22.1**	**2.9**	**78.5**	**1.3**	**1465**	**48.0**
	TA-ASF	**22.1**	**2.9**	**79.7**	**0.1**	**1465**	**48.0**
DeiT-B	Baseline [4]	86.6	17.6	81.8	0.0	300	0.0
	DynamicViT [27]	89.4	11.5	81.4	0.4	454	51.3
	IA-RED^2 [29]	86.6	11.6	80.9	0.9	453	51.1
	Evo-ViT [28]	86.6	11.6	81.3	0.5	438	46.0
	EViT [26]	86.6	11.6	81.3	0.5	445	48.3
	TA-ASF(off-the-shelf)	**86.6**	**11.5**	**80.7**	**1.1**	**455**	**51.7**
	TA-ASF	**86.6**	**11.5**	**81.8**	**0.0**	**455**	**51.7**

TABLE III
Results of deploying the DeiT-T/S/B models with and without the TA-ASF method on the Jetson Orin NX and Jetson AGX Orin. Results without the TA-ASF prefix in the table represent deploying the model directly.

Device	Model	Top-1Acc (%)	FLOPs (G)	Throughput (img/s)
Orin NX	DeiT-T	72.2	1.3	203
	TA-ASF-DeiT-T	**72.1**	**0.8**	**274**
	DeiT-S	79.8	4.6	91
	TA-ASF-DeiT-S	**79.7**	**2.9**	**133**
	DeiT-B	81.8	17.6	36
	TA-ASF-DeiT-B	**81.8**	**11.5**	**51**
AGX Orin	DeiT-T	72.2	1.3	326
	TA-ASF-DeiT-T	**72.1**	**0.8**	**440**
	DeiT-S	79.8	4.6	157
	TA-ASF-DeiT-S	**79.7**	**2.9**	**230**
	DeiT-B	81.8	17.6	60
	TA-ASF-DeiT-B	**81.8**	**11.5**	**86**

set. In the fusion stage, the new token set is matched and fused based on similarity scores, with weighted combinations forming new tokens.

Experimental results show that each module improves the model's accuracy to varying degrees under different computational budgets. The combination of both strategies yields the best results, especially under low computational budgets, with improvements of up to 1.3%. In other scenarios, the improvements are 0.7% and 0.3%, respectively. It is evident

TABLE IV
Comparison with different variants of ViT on ImageNet. We use LV-ViT-S as the base model for compression.

Method	Top-1 Acc(%)	FLOPs(G)	Params(M)
ViT-Base [1]	77.9	17.6	86.6
DeiT-S [4]	79.8	4.6	22.1
DeiT-B [4]	81.8	17.6	86.6
PVT-Small [38]	79.8	3.8	24.5
PVT-Medium [38]	81.2	6.7	44.2
CPVT-Small-GAP [44]	81.5	4.6	23.0
RegNetY-8G [39]	81.7	8.0	39.0
RegNetY-16G [39]	82.9	16.0	84.0
CrossViT-S [40]	81.0	5.6	26.7
CrossViT-B [40]	82.3	14.1	64.1
Swin-T [11]	81.3	4.5	29.0
Swin-S [11]	83.0	8.7	50.0
Swin-B [11]	83.3	15.4	88.0
T2T-ViT-14 [5]	81.5	4.8	21.5
T2T-ViT-19 [5]	81.9	8.5	39.2
T2T-ViT-24 [5]	82.3	13.8	64.1
Cvt-13 [41]	81.6	4.5	20.0
CoaT-Small [42]	82.1	12.6	22.0
CoaT-Mini [42]	81.0	6.8	10.0
TNT-S [43]	81.5	5.2	23.8
TNT-B [43]	82.9	14.1	65.6
LV-ViT-S [31]	83.3	6.6	26.2
DynamicViT-LV-S [27]	83.0	4.6	26.9
PS-LV-S [37]	82.4	4.7	26.2
EViT-LV-S [26]	83.0	4.7	26.2
TA-ASF-LV-S(off-the-shelf)	**81.7**	**4.6**	**26.2**
TA-ASF-LV-S	**83.1**	**4.6**	**26.2**

TABLE V
Results of using different compression strategies on the DeiT-S model at various computational levels.

Strategy	Top-1 Acc(%)	FLOPs(G)
DeiT-S	79.8	4.6
+Attention select	77.7	2.3
+Attention aware sampling	77.9	2.3
+Weighted fusion	78.5	2.3
Attention aware sampling+Weighted fusion	79.0	2.3
+Attention select	78.8	2.6
+Attention aware sampling	79.0	2.6
+Weighted fusion	79.3	2.6
Attention aware sampling+Weighted fusion	79.5	2.6
+Attention select	79.4	2.9
+Attention aware sampling	79.5	2.9
+Weighted fusion	79.6	2.9
Attention aware sampling+Weighted fusion	79.7	2.9

that the lower the computational budget, the more effective our method becomes. This is because our sampling and fusing modules consider the global impact of low-importance tokens and efficiently merge similar tokens, allowing the model to retain more effective tokens. Even with a significantly reduced number of tokens, the remaining ones contribute greatly to the task, minimizing the negative impact on performance. This enables excellent performance even under very low computational budgets, further proving the superiority of considering both types of redundancy in the model. We also observed that the fusion module alone outperforms the sampling module alone, as fusion preserves more information compared to direct token discarding, particularly under low computational budgets. This insight could be beneficial for future researchers.

2) Different Token Fusion Strategies: As shown in Table VI, we compared our method with two common fusion strategies to evaluate its effectiveness. i) Average Fusing Strategy: Similar tokens are replaced with the average value of the two tokens. ii) Max Fusing Strategy: For two similar tokens, the larger value is retained, and the other part is discarded.

Experimental results show that our method consistently achieves the best performance across models of different scales. Specifically, in the DeiT-B model, our strategy improves accuracy by 0.5% compared to other fusion strategies, and it also achieves improvements of $0.2\% \sim 0.4\%$ in other models. These results strongly demonstrate the effectiveness of our strategy.

V. Related Work

To accelerate Vision Transformers (ViTs), researchers have proposed various methods, including Efficient architecture design, Quantization on ViTs, and Token compress techniques.

Efficient architecture design: ViT [1] is the first work to directly apply the Transformer architecture to image classification tasks. By dividing an image into fixed-size patches and processing them through multiple Transformer layers to capture global relationships between tokens, ViT demonstrated

TABLE VI
Results on DeiT-T/S/B model using different token fusion strategies.

Model	Strategy	Top-1 Acc(%)	FLOPs(G)
DeiT-T	Average fusing	71.7	0.8
	Max fusing	71.7	0.8
	Ours	**72.1**	**0.8**
DeiT-S	Average fusing	79.5	2.9
	Max fusing	79.4	2.9
	Ours	**79.7**	**2.9**
DeiT-B	Average fusing	81.3	11.5
	Max fusing	81.2	11.5
	Ours	**81.8**	**11.5**

superior performance over traditional Convolutional Neural Networks (CNNs) on large-scale datasets like ImageNet [32]. However, ViT relies heavily on large-scale datasets. To address this dependency, DeiT [4] introduces a distillation token and adjusts training optimization strategies, improving ViT's training efficiency and achieving competitive performance on ImageNet without the need for extensive pre-training. MAE [23] enhances model representation and generalization by training with partially masked input images, offering a new method for improving model robustness and handling sparse data. DeepViT [25] further improves ViT's depth, excelling in image classification tasks while exploring various compression strategies. It highlights the potential of deep Transformers to achieve state-of-the-art results in computer vision and demonstrates the potential of deep models to enhance performance and computational efficiency.

Despite the success of ViT and its subsequent studies, demonstrating its strong potential as an alternative to CNNs, high computational costs remain a significant challenge for deployment at edge devices. In this paper, we focus on accelerating existing ViT models by reducing the number of tokens.

Quantization on ViTs: Quantization reduces the computational and storage requirements of machine learning models by converting model parameters and computations from high precision (e.g., 32-bit floating point) to low precision (e.g., 8-bit integers). This technique significantly reduces the memory footprint and computational complexity of models. PTQ-ViT [45] proposes an effective post-training quantization algorithm that determines the bit-widths of each layer by analyzing the relationship between quantization loss and feature diversity in different layers, using kernel norms of attention maps and output characteristics in Transformer layers to reduce the memory and computational costs of Vision Transformers. FQ-ViT [46] introduces an extreme quantization method that quantizes not only linear layers and self-attention layers but also uses power-of-two factors and Log2 quantizers for LayerNorm and SoftMax. At ultra-low precision (e.g., 4-bit integers), Post-training quantization (PTQ) suffers significant performance degradation due to the inability to optimize large quantization errors with limited calibration images, highlighting the urgent need for Quantization-aware training (QAT) to achieve more

accurate low-bit Vision Transformers. Q-ViT [47] identifies MSA and GELU as highly sensitive to quantization and proposes a fully differentiable quantization method that optimizes quantization scales by dynamically adjusting the bit-widths of heads in ViT. Quantformer [48] proposes considering self-attention levels in quantized Transformers to maintain consistency between quantized Vision Transformers and full-precision models.

Although quantization operations can effectively reduce the computational load of models, not all hardware supports low-precision computations. Therefore, quantized models need to run on specific hardware, significantly limiting their deployment range. Additionally, both PTQ and QAT require retraining and cannot be directly applied to existing models.

Token compression: In the field of vision transformers, token compression primarily refers to a specific method aimed at reducing the number of input tokens in the Transformer encoder. Inspired by [2], we understand that the high computational cost of standard transformers mainly arises from the quadratic time complexity of multi-head self-attention (MHSA). Consequently, various methods have emerged to reduce the number of input tokens based on this finding. These methods can be categorized into two groups according to the types of redundancy they address.

The first group, focusing on reducing quantity redundancy, DynamicViT [27] inserts an additional lightweight predictive module between different layers to score and discard unimportant tokens; IA-RED^2 [29] introduces an interpretable module to dynamically remove tokens irrelevant to the input; Evo-ViT [28] selects informative and uninformative tokens by leveraging the global class attention mechanism of Vision Transformers, allowing for slow updates for informative tokens and fast updates for uninformative ones; A-ViT [49] further designs an adaptive token pruning mechanism based on class token attention to dynamically adjust compression ratios. This method of discarding unimportant tokens significantly reduces model complexity but risks neglecting correlations between adjacent tokens, potentially harming model performance.

The second group of methods primarily focused on reducing similar redundancy, Tome [30] divides all input tokens into two sets using a soft-matching algorithm and then performs one-to-one or many-to-one fusion based on the similarity of tokens within each set. While this straightforward method of merging similar tokens can rapidly reduce the token count, it overlooks the crucial contribution of different tokens to the task. In contrast, our method considers both types of redundancy, taking into consideration the relevance and importance of tokens, and has demonstrated strong performance in experiments.

VI. Conclusions

In this paper, we propose an efficient token compression method, TA-ASF, that can more aggressively compress vision transformers. We fully consider the two types of redundancy present in the transformer input sequence and propose two different algorithms to address them. For unimportant tokens, we do not discard all of them directly but selectively discard a portion. For similar tokens, we adopt an efficient fusion strategy. This method significantly enhances the retention of effective information while reducing the number of tokens, thereby avoiding significant performance degradation. Our method is simple and effective, requiring no additional trainable parameters. It can be added as a plug-and-play module to existing pre-trained vision transformers to reduce their FLOPs without any additional training. Experimental results show that our approach significantly improves the performance of edge devices while maintaining the top-1 accuracy of the model. Specifically, for the DeiT-B model, FLOPs are reduced by 34.7%, and throughput increases by 1.41 $\sim$ 1.52 times. For the DeiT-S model, FLOPs are reduced by 37%, and throughput increases by 1.46 $\sim$ 1.48 times, with a slight accuracy reduction of 0.1%. Similar results are achieved on LV-ViT-S, further demonstrating the effectiveness of our method.

VII. Acknowledgment

We are grateful to anonymous reviewers for their constructive suggestions. We sincerely thank Ms. Xiaohong Wang for supporting the research and Dr. Liangkai Liu for valuable suggestions on writing. This work is supported in part by the National Natural Science Foundation of China under Grant No. 62402475, No. 62162067 and No. 62101480, in part by the Yunnan Province expert workstations under Grant202305AF150078, in part by Yunnan Fundamental Research Projects under Grant Nos. 202401AT070474.

References

[1] 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.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "Levit: a vision transformer in convnet's clothing for faster inference," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 259–12 269.

[4] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[5] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 558–567.

[6] Y. Cao, Y. Fan, J. Bin, and Z. Liu, "Lightweight transformer for multi-modal object detection (student abstract)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, 2023, pp. 16 172–16 173.

[7] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," in *European conference on computer vision*. Springer, 2022, pp. 280–296.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[9] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[10] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in neural information processing systems*, vol. 34, pp. 12 077–12 090, 2021.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[12] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International conference on machine learning*. PMLR, 2020, pp. 1691–1703.

[13] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[14] A. Y. Ding, E. Peltonen, T. Meuser, A. Aral, C. Becker, S. Dustdar, T. Hiessl, D. Kranzlmüller, M. Liyanage, S. Maghsudi *et al.*, "Roadmap for edge ai: A dagstuhl perspective," pp. 28–33, 2022.

[15] X. Zhang, M. Qiao, L. Liu, Y. Xu, and W. Shi, "Collaborative cloud-edge computation for personalized driving behavior modeling," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing(SEC)*. IEEE, 2019, pp. 209–221.

[16] A. Rahmanian, A. Ali-Eldin, S. K. Tesfatsion, B. Skubic, H. Gustafsson, P. Shenoy, and E. Elmroth, "Ravas: Interference-aware model selection and resource allocation for live edge video analytics," in *2023 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2023, pp. 27–39.

[17] A. Xie and Y. Peng, "Improving the quality of inference for applications using chained dnn models during edge server handover," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. IEEE, 2022, pp. 516–520.

[18] Y. Rao, J. Lu, J. Lin, and J. Zhou, "Runtime network routing for efficient image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2291–2304, 2018.

[19] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[20] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[21] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.

[22] S. Mehta and M. Rastegari, "Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[23] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[24] H. Cai, J. Li, M. Hu, C. Gan, and S. Han, "Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 302–17 313.

[25] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng, "Deepvit: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886*, 2021.

[26] 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.

[27] Y. Rao, Z. Liu, W. Zhao, J. Zhou, and J. Lu, "Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10 883–10 897, 2023.

[28] Y. Xu, Z. Zhang, M. Zhang, K. Sheng, K. Li, W. Dong, L. Zhang, C. Xu, and X. Sun, "Evo-vit: Slow-fast token evolution for dynamic vision transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 2964–2972.

[29] 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.

[30] D. Bolya, C. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[31] Z.-H. Jiang, Q. Hou, L. Yuan, D. Zhou, Y. Shi, X. Jin, A. Wang, and J. Feng, "All tokens matter: Token labeling for training better vision transformers," *Advances in neural information processing systems*, vol. 34, pp. 18 590–18 602, 2021.

[32] 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*. Ieee, 2009, pp. 248–255.

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

[34] 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.

[35] 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 latency-aware soft token pruning," in *European conference on computer vision*. Springer, 2022, pp. 620–640.

[36] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing sparsity in vision transformers: An end-to-end exploration," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 974–19 988, 2021.

[37] Y. Tang, K. Han, Y. Wang, C. Xu, J. Guo, C. Xu, and D. Tao, "Patch slimming for efficient vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 165–12 174.

[38] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 568–578.

[39] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 428–10 436.

[40] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 357–366.

[41] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 22–31.

[42] W. Xu, Y. Xu, T. Chang, and Z. Tu, "Co-scale conv-attentional image transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9981–9990.

[43] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Advances in neural information processing systems*, vol. 34, pp. 15 908–15 919, 2021.

[44] X. Chu, Z. Tian, B. Zhang, X. Wang, and C. Shen, "Conditional positional encodings for vision transformers," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[45] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-training quantization for vision transformer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 092–28 103, 2021.

[46] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, "Fq-vit: Post-training quantization for fully quantized vision transformer," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, Ed. ijcai.org, 2022, pp. 1173–1179.

[47] Z. Li, T. Yang, P. Wang, and J. Cheng, "Q-vit: Fully differentiable quantization for vision transformer," *arXiv preprint arXiv:2201.07703*, 2022.

[48] Z. Wang, C. Wang, X. Xu, J. Zhou, and J. Lu, "Quantformer: Learning extremely low-precision vision transformers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, pp. 8813–8826, 2023.

[49] 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.