Applied Intelligence (2025) 55:342
https://doi.org/10.1007/s10489-025-06265-z

Automatic pruning rate adjustment for dynamic token reduction in vision transformer

Ryuto Ishibashi[1] · Lin Meng[2]

Accepted: 4 January 2025 / Published online: 18 January 2025

Abstract
Vision Transformer (ViT) has demonstrated excellent accuracy in image recognition and has been actively studied in various fields. However, ViT requires a large matrix multiplication called Attention, which is computationally expensive. Since the computational cost of Self-Attention used in ViT increases quadratically with the number of tokens, research to reduce the computational cost by pruning the number of tokens has been active in recent years. To prune tokens, it is necessary to set the pruning rate, and in many studies, the pruning rate is set manually. However, it is difficult to manually determine the optimal pruning rate because the appropriate pruning rate varies from task to task. In this study, we propose a method to solve this problem. The proposed pruning rate adjustment adjusts the pruning rate so that the training loss is converged by Gradient-Aware Scaling (GAS). In addition, we propose Variable Proportional Attention (VPA) for Top-K, a general-purpose token pruning method, to mitigate the performance loss due to pruning. For the CIFAR-10 dataset, several competitive pruning methods improve recognition accuracy over manually setting the pruning rate; eTPS+Adjust on Hybrid ViT-S achieves 99.01% Accuracy with -31.68% FLOPs. Furthermore, Top-K+VPA outperforms token merging when the pruning rate is large for trained ViT-L inference on ImageNet-1k and has superior scalability in the Accuracy-Latency relation. In particular, when Top-K+VPA is applied to ViT-L on a GPU environment with a pruning rate of 6%, it achieves 80.62% Accuracy on the ImageNet-1k dataset with -50.44% FLOPs and -46.8% Latency.

Keywords Vision transformer · Image recognition · Token reduction · Pruning rate adjustment

1 Introduction

Deep learning models for image recognition and natural language processing have been further researched with the advent of Transformer [1]. In image recognition, the Vision Transformer (ViT) [2–4] outperforms traditional Convolutional Neural Networks (CNNs) and has applications in a variety of fields [5–7]. ViT is a model that applies the Transformer, developed as a natural language processing (NLP) model by treating patches of segmented input images as tokens to image recognition tasks. ViT can recognize images by dividing the input image into patches and treating each patch as a token, just as the Transformer performs with NLP. The Transformer extracts various information from input tokens through the mechanism called Attention, which aggregates and reflects dependencies among tokens. Due to ViT's superior scalability, the performance potential of ViT is much higher than that of CNNs when the model is scaled up. However, the high computational complexity of large ViTs limits their practicality.

To alleviate this problem, reducing the number of tokens is an efficient way to reduce the computational complexity since the computational complexity of the Transformer's Attention grows as the square of the number of input tokens. On the other hand, the method of selecting tokens to be removed is very important because token reduction causes a non-negligible loss of information. Token reduction and token merging exist as methods to reduce the number of tokens and are very hot topics for reducing ViT calculations. In the case of token reduction, many studies delete tokens that are not important for classification, and the most common method is

✉ Lin Meng
menglin@fc.ritsumei.ac.jp

Ryuto Ishibashi
ri0097fx@ed.ritsumei.ac.jp

1 Graduate School of Science and Engineering, Ritsumeikan University, Shiga, Japan

2 College of Science and Engineering, Ritsumeikan University, Shiga, Japan

called Top-K, which retains the top k tokens that have a high contribution to the class tokens, as shown in Fig. 1. Since the contribution to the class token is earned naturally in the Attention calculation, little additional calculation is required. When merging tokens, it is common practice to merge similar tokens, and the method can mitigate missing information. Although these methods are very useful for token pruning, they all require manual setting of the number of tokens to be pruned. It is known that a higher-performance model can be built by selecting an appropriate pruning rate for each layer [8], and the search for a better pruning rate is one of the important themes in token pruning. However, since searching for the optimal pruning rate requires multiple re-trainings, which in turn increases learning time, research on automatic pruning rate optimization is valuable.

In this paper, we propose a pruning method that automatically adjusts the pruning rate while training the model, and build a flow that gradually adjusts the pruning rate while cycling through the target layers. The proposal uses the variation of training loss and the variation of gradient in MLP to adjust the pruning rate layer by layer. The gradient-based method gradually reduces the variability of training loss and searches for a pruning rate such that the training loss converges. In addition, to mitigate ViT recognition accuracy loss due to token reduction, we also extend the idea of Proportional Attention [9] and propose VPA that can be applied to the Top-K method. The major contributions of this research are as follows.

- Propose a method that automatically adjusts the pruning rate according to the training loss convergence, and show that this proposal outperforms the manual setting method for several token pruning methods that require manual setting of the pruning rate
- Variable Proportional Attention (VPA), which modifies Proportional Attention to apply to Top-K and introduces variable temperature, not only improves Top-K performance in inference with ImageNet-1k using off-the-shelf ViT-L but also outperforms token merging methods when pruning rates are high
- In ViT-L, compared to the baseline model, Top-K+VPA achieves -46.8% Latency on GPU, -42.0% on GPU-TRT, and 34.9% on CPU, better Accuracy-latency scalability than other competitive pruning methods

2 Related works

2.1 Vision transformer

Vision Transformer (ViT) [2] is a Transformer-based deep learning model that has been extensively studied in recent

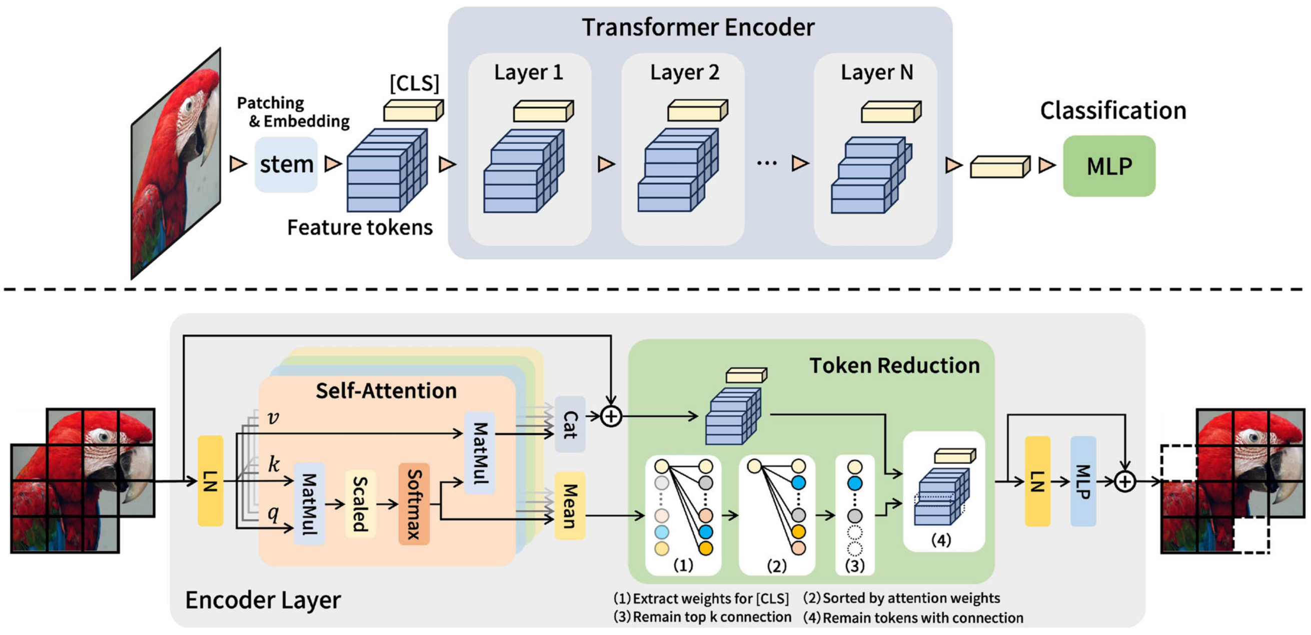

Fig. 1 Overview of Top-K method for Transformer Encoder: The Top-K policy keeps the top k tokens with the highest contribution to the `[CLS]` in each layer. Token removal is done after ViT's self-attention and is performed in the following steps. (1) extract attention weights for the `[CLS]` (2) sorted by attention weights (3) remain top k connection (4) remain tokens with remained connection. The attention weights of self-attention are used as the contribution to the `[CLS]`, and the average value of each head of Multi-head attention is used as the significance score $\mathcal{S}$ as shown in (3)

years and is an extension of a NLP model called BERT[10] for image recognition. In ViT, the input image is divided into patches and treated as input tokens, and the dependency of the patches is reflected in its dedicated token named class token `[CLS]`. Then, only `[CLS]` is extracted and passed through MLP for classification.

The attention is formulated as in (1) and is calculated by weighting the dependency between query and key to value where the function $\psi(\cdot)$ is the softmax function. In the case of self-attention, the input vector is linearly transformed into Q, K, and V.

$$\text{Attention}(Q, K, V) = \psi\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Each layer of the Transformer Encoder consists of Layer Normalization, Multi-Head Attention, and MLP. In Multi-Head Attention, the input vector is divided into the number of heads and Scaled Dot-Product Attention as shown in (1) is calculated for the linearly transformed vector into Query, Key, and Value. There is a lot of research going on to improve ViT, including research on effective input tokens [11], changes in positional encoding [12], and the use of hierarchical structures [13].

2.2 Efficient token reduction

Numerous studies attempt to improve ViT performance by dynamically reducing the number of patches because ViT works with an arbitrary number of patches (tokens).

In this subsection, studies that merge tokens and studies that remove tokens are described.

2.2.1 Token merging

Research on merging tokens has begun to increase in recent years, and various methods have been proposed for different purposes, such as improving accuracy, reducing computational complexity, and improving latency. In token merging, many studies discuss how to merge tokens and which tokens to refer to. EViT [14] is an extension of Top-K that preserves information by merging the tokens to be reduced into one token and generating additional tokens. eTPS [15] is a modified version of EViT that merges the reduced tokens into the most similar token held, rather than combining them into a single token. GroupViT [16] groups tokens by cross-attention of learnable tokens and image tokens to achieve high resolution and density. ToMe [9], which uses lightweight soft bipartite matching to merge similar tokens and achieves twice the throughput without re-training with a slight accuracy loss.

2.2.2 Token reduction

Among the research efforts to dynamically reduce the number of tokens in Transformer, the removal of unnecessary tokens has been popular [17, 18]. Some attempt to remove tokens completely and others to keep less important tokens by masking, etc., although the latter is adopted in most cases. This is because, although the Transformer can dynamically prune tokens and operate with an arbitrary number of input tokens, the number of tokens retained varies depending on the sentence or image, which results in the loss of parallelism during batch processing. To completely remove tokens while maintaining the parallelism of batch processing, the number of tokens to be pruned must be fixed regardless of the input, and the pruning rate must be set manually.

The method of determining which tokens to prune is also very important for token reduction: some methods based on Transformer's Self-Attention score use the L1/L2 norm or the activation value, while IA-RED^2 [19] uses the Multi-head Interpreter and completely removes anything below the threshold value for the norm of attention weight. Adaptive Token Sampling (ATS) [20] uses the product of attention weights $\mathcal{A}_{c,j}$ to class tokens and the norm of value $\|\mathcal{V}_j\|$ to compute importance scores, and keeps top k tokens where each index indicates the number of tokens $j \in \{1, \ldots, N\}$ when the number of tokens is N. In the window-based self-attention used in the Swin Transformer, SparseViT [21] uses activation sparsity to prune with less activation L2 norm per window.

2.2.3 Proportional attention

Reducing tokens with smaller contributions reduces the bias among attention weights, which could negatively affect performance by flattening the distribution. To address this issue, ToMe [9] adopts Proportional Attention, which adds a bias term $\log s$ to Self-Attention as (2).

$$\mathcal{A} = \psi\left(\frac{QK^T}{\sqrt{d_k}} + \log s\right)V \tag{2}$$

where s is a row vector that tracks the number of merged tokens for each token, and weighting according to the number of merged tokens yields the same softmax attention as when replicating a similar key. The row vector s of Proportional Attention is used to weight each token proportionally, with the sum of s equal to the number of tokens before pruning. Proportional attention requires information for similar tokens by calculating bipartite matching algorithms using the Key of Self-Attention to get the vector s.

2.2.4 Pruning rate of each layer

X-Pruner [8] considers the explainability of pruning and achieves a significant reduction in the computational complexity of ViT by pruning units such as the head and the linear layer in the feed-forward process during inference by using differentiable masks. It is described that setting the pruning rate manually is a problem and they propose the method for learning thresholds for each layer by calculating the loss between the target pruning rate and the current pruning rate using an extended Lagrangian method. It is also noted within that paper that the choice of pruning rate for each layer has a significant impact on the final performance.

Algorithm 1 Self-attention with token reduction.

```
Input: $x$: input tokens, $N$: number of tokens,
1: $W_q$, $W_k$, $W_v$: weights for query, key, and value,
2: $d_k$: dimension of key, $r^l$: pruning rate of layer $l$,
3: $s$: row vector for Proportional Attention,
4: $H$: number of heads
Output: $x_{\text{prune}}$ : pruned tokens
5: $q, k, v = W_q x, W_k x, W_v x$
6: $\mathcal{A} = \psi(qk^T/\sqrt{d_k} + \log s)$
7: $x \leftarrow \mathcal{A}v$
8: $k = N - \lfloor r^l N \rfloor$
9: $\hat{\mathcal{A}}_{c,:} = 1/H \sum_h \mathcal{A}^h_{c,:}$
10: $\hat{\mathcal{A}}_{c,c} \leftarrow \infty$                ▷ make score of [CLS] token top
11: index = argsort($\hat{\mathcal{A}}_{c,:}$)
12: source_index ← index[..., : $k$]               ▷ extract top $k$ indices
13: $x_{\text{prune}} \leftarrow$ gather($x$, source_index)
```

3 Methodology

Although the optimal pruning rate varies from task to task, many pruning methods involve non-differentiable processes and hence the pruning rate cannot be directly optimized. Therefore, we propose a pruning rate adjustment method that can be applied to non-differentiable pruning processes. The proposed method adjusts the pruning rate using the variation in loss and the norm change in the MLP gradient of each layer.

3.1 Dynamic token reduction

Figure 1 shows Top-K, the commonly used token pruning method. In the Top-K method, to dynamically prune tokens with low importance, the dependency of each token on `[CLS]` is used, which is calculated by self-attention. Algorithm 1 shows the pseudo-code of self-attention with token reduction. $\mathcal{A}$ indicates attention weights for all tokens and and $\hat{\mathcal{A}}_{c,:}$ is attention weights for `[CLS]`. k indicates the number of remained tokens, and index and source_index are indices of sorted $\hat{\mathcal{A}}_{c,:}$ of all connection and reduced connection. The pruning algorithm is based on attention weights for `[CLS]` representing how much the token contributed to the classification. The process is performed by sorting the attention weights and retrieving connections according to the pruning rate. $\hat{\mathcal{A}}_{c,c}$ represents the weight of the `[CLS]` relative to the `[CLS]`, and setting this value to infinity always makes the class token first one when sorting. The function gather$(\cdot)$ takes out the tensor according to the given index, which is sorted, and the top k tokens are taken out from x according to the sliced index.

The Top-K method uses significance scores $\mathcal{S}$ in (3) to select tokens.

$$\mathcal{S} = \left(\hat{\mathcal{A}}_{c,1}, \hat{\mathcal{A}}_{c,2} \cdots \hat{\mathcal{A}}_{c,N}\right) \in \mathbb{R}^{1\times N} \qquad (3)$$
$$s.t. \quad \hat{\mathcal{A}}_{c,j} = \frac{1}{H}\sum_h \mathcal{A}^h_{c,j} \quad (\mathcal{A}^h \in \mathbb{R}^{N\times N})$$

where $\mathcal{A}^h$ is attention weights in h-th head and $\mathcal{A}^h_{c,j}$ indicates attention weights of a token j to the `[CLS]`.

3.2 Variable proportional attention (VPA)

The base idea of Proportional Attention is to preserve the information of the tokens to be merged in a row vector s. As the number of merged tokens increases, vector s is used to act proportionally on the soft-maxed attention map. However, this method can only be applied to pruning processes with merging and cannot be applied to processes that simply remove tokens, such as Top-K. Therefore, this paper modifies Proportional Attention, which can be applied to the Top-K-based token reduction method. Proportional Attention can be applied in general token reductions by interpreting the row vector s to preserve the information of the tokens to be removed in a weighted sum of the tokens to be kept (Fig. 2).

Variable Proportional Attention (VPA) aims to preserve information about the tokens to be removed by aggregating their similarity to the tokens to be kept. VPA has a variable temperature that varies with a sum of token similarity as shown in (4)–(5). The method uses the key in self-attention, as in the original Proportional Attention, and calculates cosine similarity matrix M between the tokens to

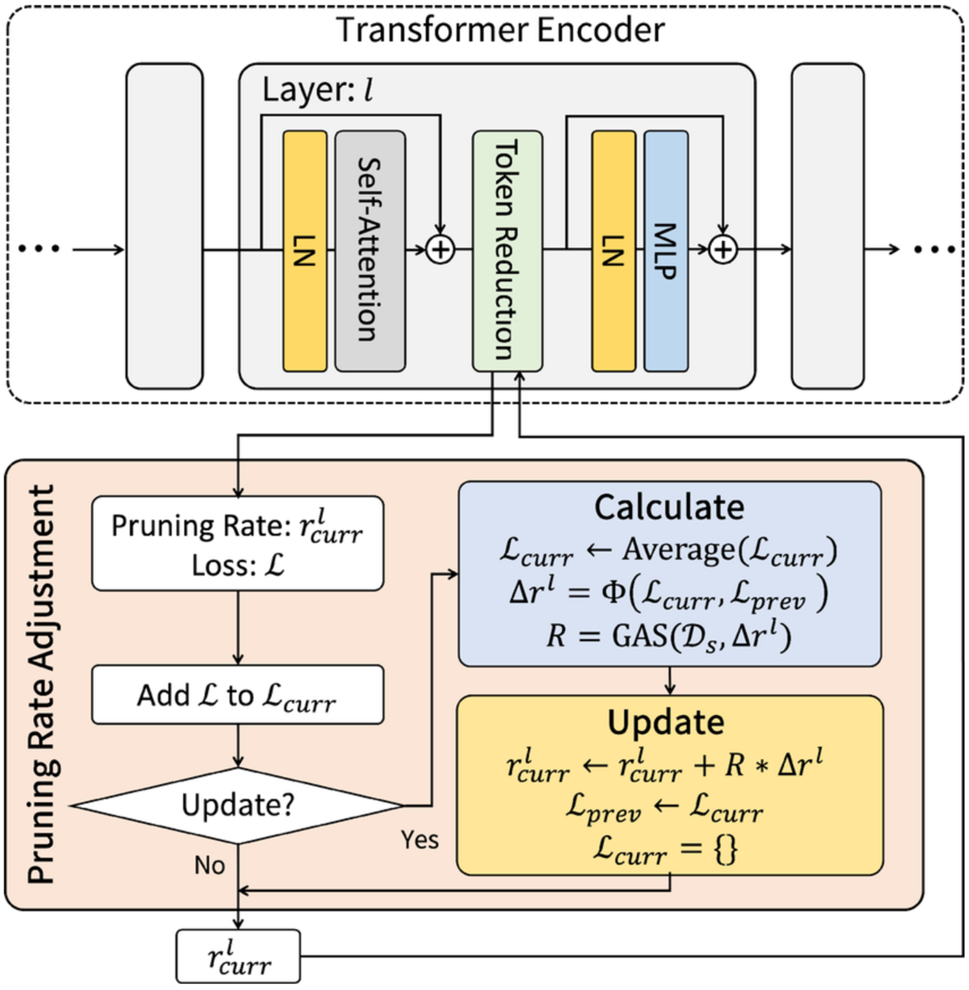
Transformer Encoder
Layer: l
LN
Self-Attention
Token Reduction
LN
MLP
Pruning Rate Adjustment
Pruning Rate: r^l_{curr}
Loss: $\mathcal{L}$
Add $\mathcal{L}$ to $\mathcal{L}_{curr}$
Update?
Yes
No
Calculate
$\mathcal{L}_{curr} \leftarrow \text{Average}(\mathcal{L}_{curr})$
$\Delta r^l = \Phi(\mathcal{L}_{curr}, \mathcal{L}_{prev})$
$R = \text{GAS}(\mathcal{D}_s, \Delta r^l)$
Update
$r^l_{curr} \leftarrow r^l_{curr} + R * \Delta r^l$
$\mathcal{L}_{prev} \leftarrow \mathcal{L}_{curr}$
$\mathcal{L}_{curr} = \{\}$
r^l_{curr}

Fig. 2 Flow of pruning rate adjustment process

be removed and those to be kept.

$$s = s_{\text{keep}} + s_{\text{rm}}\psi(M') \tag{4}$$

$$M' = \frac{M}{\left(\sum_{j=0}^{k} M_{i,j}\right) \times \tau} \tag{5}$$

where $s_{\text{keep}} \in \mathbb{R}^{1\times k}$ and $s_{\text{rm}} \in \mathbb{R}^{1\times r}$ are the row vectors of k tokens to be kept, and r tokens to be removed, respectively. τ indicates the base value of temperature. The similarity matrix with k tokens and r tokens is $M \in \mathbb{R}^{r\times k}$, and M' is the variable temperature applied to M. M' has a flat distribution when the sum of similarity is large and a sharp distribution when similarity is small. When similarity is small, the tokens tend to be more important and their information is less likely to be distributed among low-similar tokens.

3.3 Automatic adjustment of pruning rate

Many dynamic pruning methods for patches (tokens) manually set the pruning rate. This is because models that change the pruning rate in the input image cannot be batch-processed, which limits their practicality. However, since the appropriate pruning rate differs depending on each task, the manual setting requires the model to be re-trained many times while changing the pruning rate. The proposal searches the pruning rate of the model for each task by adjusting the pruning rate while training.

Each layer has its pruning rate adjusted according to the Algorithm 2. $\mathcal{C}$ is the set with all the timings to switch layers to cyclically switch layers to adjust. n represents the frequency of updating the pruning rate, updating the pruning rate r^l of the current layer according to the amount of change in loss during n epochs. Since this method stocks losses and uses the

Algorithm 2 Automatic adjustment of pruning rate.

```
Input: $L$: number of layers, $T$: epochs,
 1: $m$: number of layer-loop,
 2: $n$: frequency of $r^l$ update,
 3: $\mathcal{D}$: dataset,
Output: $r^l$: adjusted pruning rate
 4: $\mathcal{C} = \{\lfloor T/mL \rfloor \times i \mid i = 1, 2, ...mL\}$
 5: for $epoch \leftarrow 1$ to $T$ do
 6:     if $epoch \in \mathcal{C}$ then
 7:         $\mathcal{L}_{\text{list}}$ = Empty List
 8:         $l \leftarrow$ Next Layer
 9:         $r^l \leftarrow$ Pruning Rate of Current Layer
10:     end if
11:     Train One Epoch
12:     $\mathcal{L} \leftarrow$ Train Loss                          ▷ get train loss
13:     Add $\mathcal{L}$ to $\mathcal{L}_{\text{list}}$
14:     if $epoch \equiv 0 \pmod n$ then
15:         $\mathcal{L}_{current} \leftarrow$ Average of $\mathcal{L}_{\text{list}}$
16:         $\Delta r^l = \Phi(\mathcal{L}_{previous}, \mathcal{L}_{current})$
17:         $\mathcal{D}_s$ = Sampling($\mathcal{D}$)                ▷ data sampling
18:         $R$ = GAS($\mathcal{D}_s, \Delta r^l$)                   ▷ scaling
19:         $r^l = r^l + R \times \Delta r^l$
20:         $\mathcal{L}_{previous} \leftarrow \mathcal{L}_{current}$
21:     end if
22: end for
```

average of these losses when updating the pruning rate, it has some robustness against unstable losses. However, when n is small for difficult tasks, the pruning rate could be unstable and learning convergence could be slow.

3.3.1 Cyclic layer adjustment

Since the proposed method varies the pruning rate while training, varying the pruning rate for all layers could result in inconsistent parameter optimization. Therefore, the pruning rate for each layer should be adjusted progressively. In this study, the method to change the layer to be adjusted cyclically is adopted. For example, when adjusting the pruning rate of a Transformer Encoder consisting of three layers, the pruning rate of one layer should not become excessively large by changing the pruning rate from $1 \to 2 \to 3 \to 1 \to 2 \to 3 \to \cdots$. In this paper, the number of layer loops m is set to 8, and the frequency of the pruning rate update n is set to 2.

3.3.2 The amount of change in the pruning rate

The amount of change in the pruning rate Δr^l, is adjusted by the descending range of loss to suppress large fluctuations in the pruning rate at the end of the learning phase and to improve learning stability. This method sets a maximum value as $\Delta r_{max} = 0.05$ in this experiment due to the large fluctuation range of the pruning rate in the early stages of learning. Equation (6) shows the function to calculate Δr^l.

$$\Phi(\mathcal{L}_1, \mathcal{L}_2) = \Delta r_{max} \left(\frac{2}{1 + e^{-\alpha(\mathcal{L}_1 - \mathcal{L}_2)}} - 1 \right) \tag{6}$$

The function $\Phi(\cdot)$ is a logistic function as $-\Delta r_{max} \leq \Phi \leq \Delta r_{max}$. α is a parameter that controls the slope of the function and can be set to a large number to suppress the pruning rate stagnation in the middle to the end stages of the training. In this paper, $\alpha = 10$ is used.

3.3.3 Gradient-aware scaling (GAS)

Since each layer has a different impact on the loss, adjusting all layers by the same loss is not appropriate. Therefore, this paper proposes a method of scaling the amount of pruning rate updates using the gradient of each layer. Specifically, for a few data that are randomly sampled from the dataset, the amount of pruning rate updates is reduced if the amount of pruning rate updates causes a large change in gradient. However, the pruning process often includes non-differentiable processes, and the gradient cannot be calculated directly. Therefore, the gradient in the MLP layer is substituted as the grad of the layer after the pruning process. The gradient for r^l is calculated as in (7).

$$G(r^l) = \left(\frac{\partial E(\mathcal{D}_s | r^l)}{\partial w_i} \,\middle|\, w_i \in \mathbf{W}^l \right) \tag{7}$$

where r^l is the pruning rate of l-th layer, $\mathbf{W}^l$ is the MLP weights, E is the loss, and $\mathcal{D}_s$ is the sample data. Equation (7) calculates the gradient set in random sample data, therefore the new pruning rate is calculated from the gradient change before and after the pruning rate change, as shown in (8)–(10).

$$r^l_{temp} = r^l_{curr} + \Phi(\mathcal{L}_1, \mathcal{L}_2) \tag{8}$$

$$R = \text{Sigmoid}\left[- \sum_{w_i \in \mathbf{W}^l} \left(G(r^l_{curr}) - G(r^l_{temp}) \right)^2 \right] \tag{9}$$

$$r^l_{new} = r^l_{curr} + R \times \Phi(\mathcal{L}_1, \mathcal{L}_2) \tag{10}$$

where r^l_{curr} indicates the current pruning rate and r^l_{temp} indicates the temporarily updated pruning rate. Equation (9) calculates R, the factor that scales the amount of update, and R is smaller the larger the gradient change. As shown in Fig. 3 scaling allows layers more affected by pruning rate changes to have a smaller pruning rate change.

GAS uses gradient variation to scale the update range of the pruning rate. Layers that are more sensitive to the pruning process are less likely to have a larger pruning rate, resulting

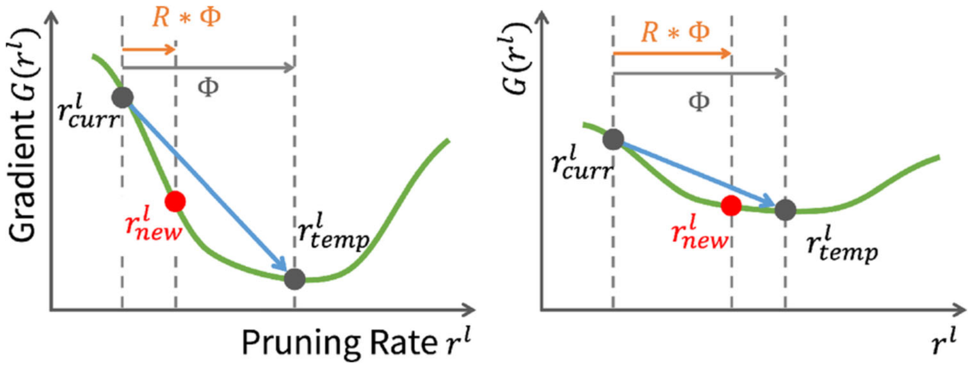

Fig. 3 Gradient-Aware Scaling: the larger the gradient of the target layer, the greater the impact of the pruning rate change, hence the smaller the amount of pruning rate updates

in a smaller pruning rate. This gradient-based method gradually reduces the variability of training loss and searches for a pruning rate such that the training loss converges.

4 Evaluation

4.1 Experimental configurations

4.1.1 Implementation conditions

Experiments in this paper are conducted on Intel(R) Core(TM) i7-13700KF CPU with a single NVIDIA GeForce RTX 4090 GPU for training, Intel(R) Xeon(R) Gold 6342 CPU with a single NVIDIA RTX A6000 GPU is used to calculate inference time, and NVIDIA Jetson Orin Nano is used for TensorRT inference. Jetson Orin Nano uses Jetpack 6.1 which contains CUDA 12.6, TensorRT 10.3, and cuDNN 9.3.

For the experimental conditions, 200 epochs of pre-trained and 200 epochs are adjusted for pruning rate adjustment, and 400 epochs are trained for all other cases. The batch size is set to 16, and AdamW is used to optimize model parameters. A warm-up cosine annealing decay is used as the scheduler for the learning rate as shown in (11)

$$lr = lr_{min} + \frac{1}{2}(lr_{max} - lr_{min})\left(1 + \cos\left(\frac{T_{curr} - W}{T_{max} - W}\pi\right)\right) \tag{11}$$

where lr_{min} and lr_{max} indicate the minimum and maximum learning rate and is set to 1×10^{-6} and $5 \times 10^{-4} \times \frac{\text{batch\_size}}{512}$ respectively, where T_{curr} indicates current epochs, T_{max} indicates the total number of training epochs, and W indicates epochs of warm-up which is set to 5.

When randomly sampling from a data set in GAS, one batch is taken from the training data and used for gradient calculation. For hyperparameters of VPA, $\tau = 0.001$ is used for vanilla ViTs, and $\tau = 0.005$ is used for Hybrid ViTs. Since the vanilla ViT has a larger number of tokens than the hybrid ViT, the τ in the hybrid model's VPA is set higher than in the vanilla model.

4.1.2 Dataset

In this experiment, CIFAR-10 and CIFAR-100 [22], Flowers [23], Oxford-IIIT Pets [24], CUB-200 [25], and Indoor67 [26] are used to fine-tune ViT models. In addition, ImageNet-1k [27] is used to evaluate the performance of pruning methods in the off-the-shell ViT model [28] without training. As data augmentation, RandomResizedCrop at 224×224, RandomHorizontalFlip, RandomErasing, and RandAugment [29] are used.

4.1.3 Comparison models

The main experiment uses the small size of vanilla ViT (ViT-S) and its hybrid model [2] to evaluate pruning methods. Each model has 12 transformer layers, 384 embedding dimensions, and 6 heads. The hybrid model has ResNet26 of ResNetv2 [30] to extract features and project to embedding dimensions. The patch size for ViT-S is 16 × 16 and the patch size for Hybrid ViT-S is effectively 32 × 32 due to CNN backbone. The number of tokens in the transformer encoder differs between the vanilla ViT and the hybrid model, and therefore the computational complexity is different. The large model of ViT (ViT-L) has 24 layers of Transformer Encoder, 1024 embedding dimensions, 16 heads, and the same patch size as ViT-S. The Hybrid ViT-L also uses ResNet50 of ResNetv2 as the CNN backbone. ResNet50 [31], ResNet10t [32], EfficientNetB3, and B6 [33] are employed as comparative models with comparable computational complexity.

4.1.4 Comparison pruning methods

In this experiment, pruning rate adjustments are made for Top-K, EViT [14], eTPS [15], and ToMe [9] as methods that require manual pruning rate settings. The first CIFAR-10 experiment is based on a 5% pruning rate, while the other dataset experiments are based on a 10% pruning rate. EViT merges the tokens to be removed and adds new tokens, however, to keep the computational complexity consistent with other methods, the pruning process reduces the number of tokens by one more. DynamicViT and A-ViT are used as pruning methods that do not require strict manual setting of the pruning rate. The target pruning rate for DynamicViT is 20% at layers 3, 6, and 9. DynamicViT and A-ViT use DeiT-S pre-trained on ImageNet-21k [34] and fine-tuned on ImageNet-1k for training, according to the official implementation. These methods follow the official documentation, with DynamicViT using pruning-free DeiT-S as a teacher for knowledge distillation and A-ViT not performing knowledge distillation. Top-K, EViT, eTPS, and ToMe are reimplemented and have customizable pruning rates. Ablation also adds Magnitude [35] and Random. Magnitude prunes tokens according to their L2 norm, keeping the top k tokens with the highest L2 norm, while Random randomly reduces tokens per layer.

4.1.5 Evaluation index

In this experiment, Top-1 Accuracy (%), FLOPs (G), and Throughput (images/s) are used as the evaluation index. Each pruning method is compared to its baseline model.

The improvement in Accuracy is measured based on the pruning method without applying the proposed method, and the improvement in FLOPs and Throughput is measured based on the vanilla model. In addition, the latency on GPU, CPU, and Jetson are measured for ViT-L to compare how the processing speed of the proposed method varies across devices. Latency measures the time to process one image by dividing the time to process one query by the batch size. When inferring on Jetson, the engine converted to TensorRT (TRT) is used to evaluate the latency of the model.

4.2 Experimental results

4.2.1 Performance comparison in CIFAR-10

Table. 1 shows results for CIFAR-10 dataset. For ViT-S, pruning rate adjustment improves Accuracy with less computation for all pruning methods. In particular, by introducing Top-K+VPA to ViT-S, we achieve 98.82% accuracy, which

Table 1 CIFAR-10 Results on ViT-S

Model	Method	Style	Prop Attn	Top-1 Accuracy (%)	FLOPs (G)	↓ (%)	Throughput (images/s)	↑ (%)
ViT-S	None			98.99	4.25	–	982	–
	Top-K	prune		98.57	3.12	26.68	1290	+31.36
	A-ViT [18]	prune		98.35	4.25	–	1067	+8.66
	DynamicViT [17]	prune		98.00	3.26	23.20	1383	+40.84
	†EViT [14]	merge		98.58	3.12	26.68	1333	+35.74
	†eTPS [15]	merge		98.59	3.12	26.68	1245	+26.78
	†ToMe [9]	merge	+PA	98.77	3.12	26.68	1276	+29.94
	Top-K+Adjust	prune		98.76 (+0.19)	3.08	27.40	1307	+33.10
	Top-K+Adjust	prune	**+VPA**	**98.82 (+0.25)**	3.08	27.40	1280	+30.35
	†EViT+Adjust	merge		98.78 (+0.20)	3.08	27.40	**1357**	**+38.19**
	†eTPS+Adjust	merge		98.76 (+0.17)	3.08	27.40	1262	+28.51
Ours	**†ToMe+Adjust**	merge	+PA	98.81 (+0.04)	**3.07**	**27.78**	1304	+32.79
Hybrid ViT-S	None			98.97	1.10	–	1109	–
	Top-K	prune		98.73	0.77	30.35	1140	+2.80
	†EViT [14]	prune		98.70	0.77	30.35	**1179**	**+6.31**
	†eTPS [15]	prune		98.69	0.77	30.35	1140	+2.80
	†ToMe [9]	merge	+PA	98.73	0.77	30.35	1140	+2.90
	Top-K+Adjust	prune		98.91 (+0.18)	0.75	31.58	1147	+3.43
	Top-K+Adjust	prune	**+VPA**	98.99 (+0.27)	**0.75**	**32.06**	1143	+3.07
	†EViT+Adjust	merge		98.87 (+0.17)	0.75	31.58	1170	+5.50
	†eTPS+Adjust	merge		**99.01 (+0.32)**	0.75	31.58	1141	+2.89
Ours	**†ToMe+Adjust**	merge	+PA	98.95 (+0.22)	0.75	31.58	1132	+2.07
CNNs	ResNet50 [31]			97.88	4.13	–	1355	–
	EfficientNet B6 [33]			98.57	3.49	–	534	–
	ResNet10t [32]			94.76	1.11	–	4813	–
	EfficientNet B3 [33]			98.23	0.96	–	1234	–

† represents a re-implementation, which allows the pruning rate to be customized

is higher than the strong token merging method. The hybrid model has far fewer FLOPs than the vanilla model since fewer tokens are input to the Transformer Encoder. In addition, the tokens in the hybrid model are refined by the CNN backbone and have many redundant tokens that are not needed for recognition. This makes the hybrid model potentially more compatible with token pruning. Therefore, with or without the proposed method, the hybrid model has fewer FLOPs than the vanilla model and has better Accuracy after the pruning method. On the other hand, the CNN backbone is also better at optimizing for small tasks; thus, training losses converge more easily, and the pruning rate adjustment method works well. As shown in Table 1, several token pruning methods show better Accuracy gains than the vanilla model, and eTPS+Adjust achieves 99.01%. Regarding Througuput, the hybrid model outperforms ViT-S. However, ViT-S has a better speedup ratio when pruning methods are introduced. The reason for this is that the hybrid model has fewer tokens input to the Transformer Encoder, and the overhead of the pruning process outweighs the speedup from the token reduction.

Regarding the CNN model, ResNet is faster and less accurate, while EfficientNet is slower and more accurate. When the pruning process is introduced to vanilla ViT, it outperforms CNNs of similar computational complexity in both Accuracy and Throughput. While the hybrid model is slower than CNNs of similar computational complexity, the accuracy of the hybrid model is significantly higher. Figure 4 shows the pruning rate for each layer of ViT-S adjusted by the proposed method. The results show that the adjusted pruning rates are very similar except for ToMe. One difference between ToMe and the other methods lies in the method used to determine the tokens to be removed. The non-ToMe methods prune tokens with small attention weights to class tokens, while ToMe preferentially merges tokens with high similarity. The MLP layer, which calculates the gradient in GAS, calculates the gradient concerning the linear projection of the remaining tokens, resulting in a difference in the adjusted token pruning rate between ToMe and the other methods.

4.2.2 Performance comparison in downstream tasks

Table 2 shows the effectiveness of the proposed method for each downstream task. In this Adj+VPA, pruning rate adjustment, and VPA are applied to Top-K. This result shows that the proposed method is not effective in some tasks. The reason could be that VPA compensates the Attention Map due to pruning. In ViT's Attention Map, background tokens have a small impact on `[CLS]`, and these are removed by token pruning. When VPA removes background tokens, it corrects the Attention Map to replicate similar background tokens to be kept, preserving information across the entire image. Thus, VPA is effective when the entire image information is needed for classification. When background information is not needed for the majority of the image, VPA could be less accurate than regular Top-K. Both the Pets and CUB-200 datasets classify animal species, and VPA is less effective for these classifications because information around the object is more important. On the other hand, in tasks where background information contributes to classification, e.g., sea background in the "ship" class, VPA can be effective and enhance the generalization performance of Top-K to more complex tasks.

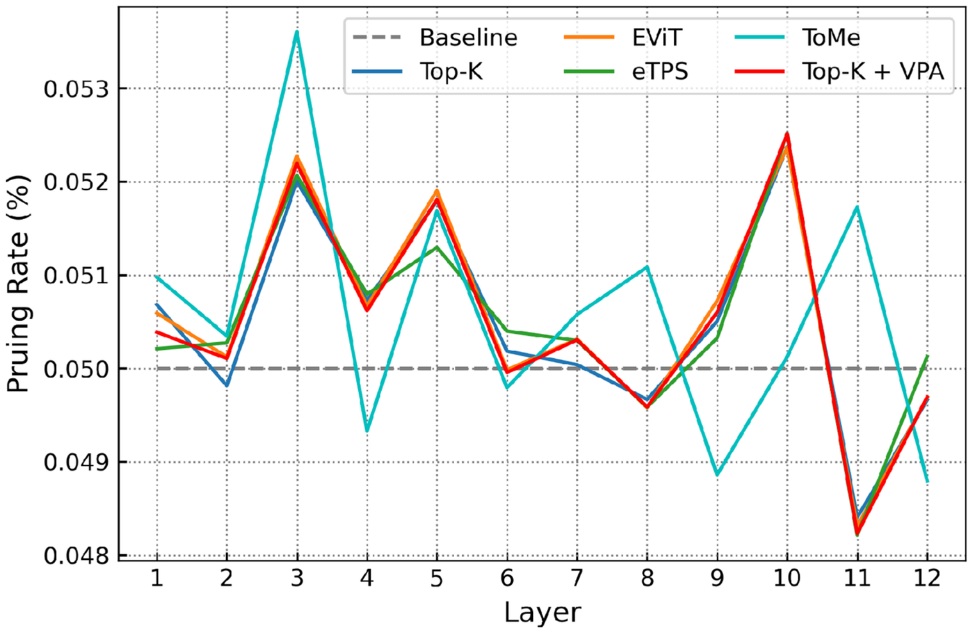

Fig. 4 Adjusted Pruning Rates in ViT-S for CIFAR-10

Table 2 Performance evaluation for classification tasks

ViT-S	Method	CIFAR-10	CIFAR-100	Flowers	Pets	CUB-200	Indoor67
Vanilla	None	99.0	91.5	99.8	93.5	86.0	86.6
	Top-K	98.4	89.3	**99.6**	**91.3**	**82.4**	82.8
	Ours*	**98.5**	**89.7**	**99.6**	90.7	81.9	**84.2**
Hybrid	None	99.0	90.9	99.8	93.4	86.3	87.4
	Top-K	98.6	89.8	**99.8**	**92.3**	**85.3**	83.1
	Ours*	**98.8**	**90.2**	**99.8**	92.2	85.0	**84.5**

* represents Adjust+VPA for Top-K method

4.3 Ablation study

4.3.1 VPA performance in ImageNet-1k

To evaluate the performance of the proposed VPA, an ablation study with the ImageNet-1k dataset is performed on an off-the-shelf ViT-L model which is pre-trained by ImageNet-21k and fine-tuned by ImageNet-1k. Table 4 shows the change in accuracy as the pruning rate varies for 6 pruning methods, including the state-of-the-art. The results in Table 4 show that the VPA improves the performance of the base Top-K at all pruning rates. Compared to ToMe, Top-K+VPA has better accuracy when the pruning rate is 6% or higher, and VPA is more effective with higher token reductions. Regarding the token merge method, the generated tokens differ from the original data, which could result in lower Accuracy than Top-K without additional training.

Table 3 compares inference time when pruning methods are introduced to ViT-L for each device. Note that ToMe cannot be converted to TensorRT because it contains functions not supported by NVIDIA's TensorRT library. In contrast to the experiment in Table 3, the process of reducing one extra token is not performed to evaluate the performance of the original EViT. For the proposed Top-K+VPA, the speedup benefit is smaller than for other models during TensorRT optimization. This is because the computation of the similarity matrix M in (5) of the tokens in the VPA is difficult to optimize. The computation of the similarity matrix involves calculating the cosine similarity between the tokens to be kept and the tokens to be removed. These token selections are difficult to optimize because they use an irregular index based on Attention Score.

Figure 5 also shows the results of the trade-offs between Accuracy, Latency, and FLOPs for each pruning rate based on the additional experiments in Table 4. The result shows that for a larger pruning rate r^l, the relationship between Accuracy and FLOPs achieves higher accuracy than the strong pruning method with almost no change in Latency, resulting in a better trade-off between computational complexity and recognition accuracy than the conventional method.

Pruning in the hybrid model outperforms it in terms of speed and accuracy; however, pruning in Edge-TRT provides little speedup. This is due to the small number of tokens in the hybrid model and the difficulty in obtaining speedups in the CNN backbone. In addition to the small number of tokens in the hybrid model, which makes pruning ineffective,

Table 3 Device-wise Inference Time for ImageNet-1k in ViT-L

Method (r^l=6%)	Top-1 Acc. (%)	GPU Latency (ms)	GPU-TRT Latency (ms)	CPU Latency (ms)	Edge-TRT Latency (ms)
(Vanilla)	84.32	8.35	3.43	46.51	55.70
Top-K	80.15	4.53 (−46.9%)	1.85 (−46.1%)	29.76 (−36.0%)	41.32 (−25.8%)
EViT	78.53	4.83 (−43.3%)	2.03 (−40.8%)	31.77 (−31.7%)	41.62 (−25.3%)
eTPS	79.26	4.68 (−45.1%)	1.94 (−43.4%)	30.75 (−33.9%)	43.96 (−21.1%)
ToMe	80.28	4.64 (−45.6%)	–	30.08 (−35.3%)	–
Top-K+VPA	**80.62**	4.54 (−46.8%)	1.99 (−42.0%)	30.27 (−34.9%)	44.14 (−20.8%)
(Hybrid)	84.07	2.94	1.71	22.80	50.73
Top-K	79.54	2.15 (−26.9%)	1.40 (−18.1%)	18.62 (−18.3%)	50.04 (−1.4%)
EViT	80.23	2.46 (−16.3%)	1.54 (− 9.9%)	20.37 (−10.7%)	50.22 (−1.0%)
eTPS	78.70	2.21 (−24.8%)	1.43 (−16.4%)	18.74 (−17.8%)	52.30 (+3.1%)
ToMe	**81.90**	2.19 (−25.5%)	–	18.69 (−18.0%)	–
Top-K+VPA	81.37	2.18 (−25.9%)	1.46 (−14.6%)	18.97 (−16.8%)	52.26 (+3.0%)

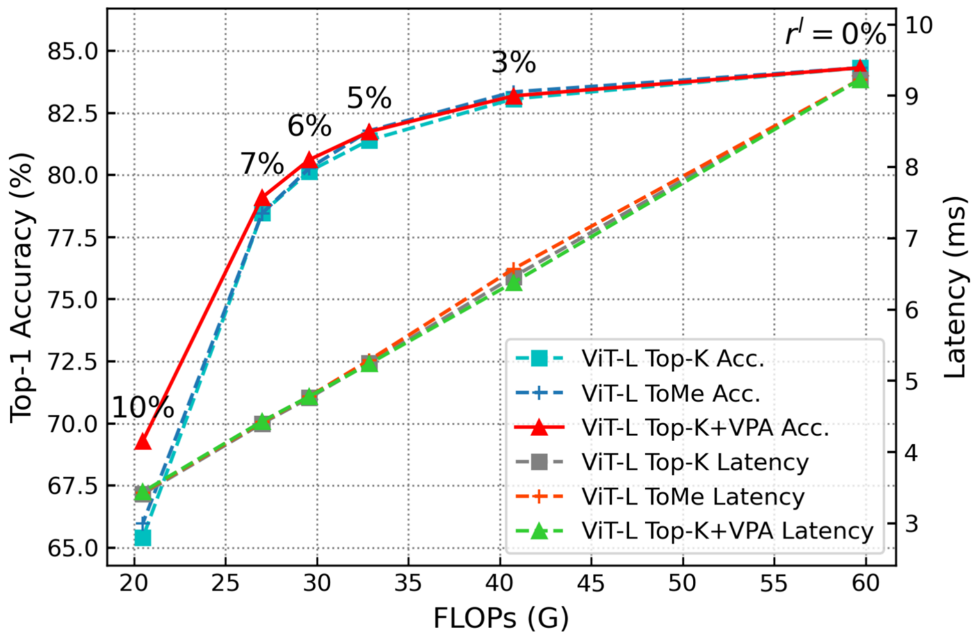

Fig. 5 Accuracy and latency for FLOPs in ViT-L pruning

the CNN backbone includes difficult-to-pruning processes such as standardized convolution and 1x1 convolution, which benefit less from TensorRT than vanilla ViT.

4.3.2 Visualization

Figure 6 is a visualization of Top-K, Top-K+VPA, Magnitude, EViT, eTPS, and ToMe in ImageNet-1k. Top-K preferentially deletes background tokens, whereas Top-K+VPA sparsely keeps tokens throughout the image in addition to object tokens. This is because the VPA aggregates the tokens to be removed into the background tokens to be left behind, resulting in a higher relative Attention Score. This sparsity allows for uniform preservation of token information across the entire image and mitigates the performance degradation caused by Top-K. This allows VPA to perform as well as or better than the token merging method, especially when pruning rates are high. Since the token reduction process is more general than token merging, VPA is highly scalable and is expected to be implemented in a variety of methods. In addition, the Magnitude-based method could remove important tokens, resulting in a significant performance loss at high pruning rates, as shown in Table 4. EViT merges the tokens to be reduced into one, thus merging tokens that are not similar. eTPS merges the tokens to be removed into the most similar token. However, eTPS does not have Proportional Attention and is inferior in Accuracy to Top-K without training, as shown in Table 4.

Table 4 Accuracy comparison in ImageNet-1k for pruning rate (ViT-L)

Method	Pruning Rate r^l			
	5%	6%	7%	10%
Top-K	81.41	80.15	78.37	65.40
EViT	78.98	77.61	75.52	60.73
eTPS	80.87	79.26	77.30	61.90
ToMe	**81.79**	80.28	78.46	65.98
Random	79.40	77.25	74.57	57.46
Magnitude	79.00	75.78	71.23	44.27
Top-K+VPA	81.75	**80.62**	**79.10**	**69.29**
FLOPs ↓(%)	44.94	50.44	54.77	65.69

5 Discussion

5.1 Proposal effectiveness

The pruning rate adjustment method refers to the training loss to search for a pruning rate such that the training loss converges. Since the hybrid model acquires the locality bias of feature extraction by the CNN backbone, training loss is more easily converged in small tasks. In the results in Table 1, 2, we can consider that the hybrid model has a larger performance gain with Adjust than the vanilla model because the training loss is easier to converge than the vanilla model. In VPA, the original Proportional Attention is extended to apply to the token reduction process, introducing a variable temperature that varies how much information is distributed among other tokens depending on the sum of similarities

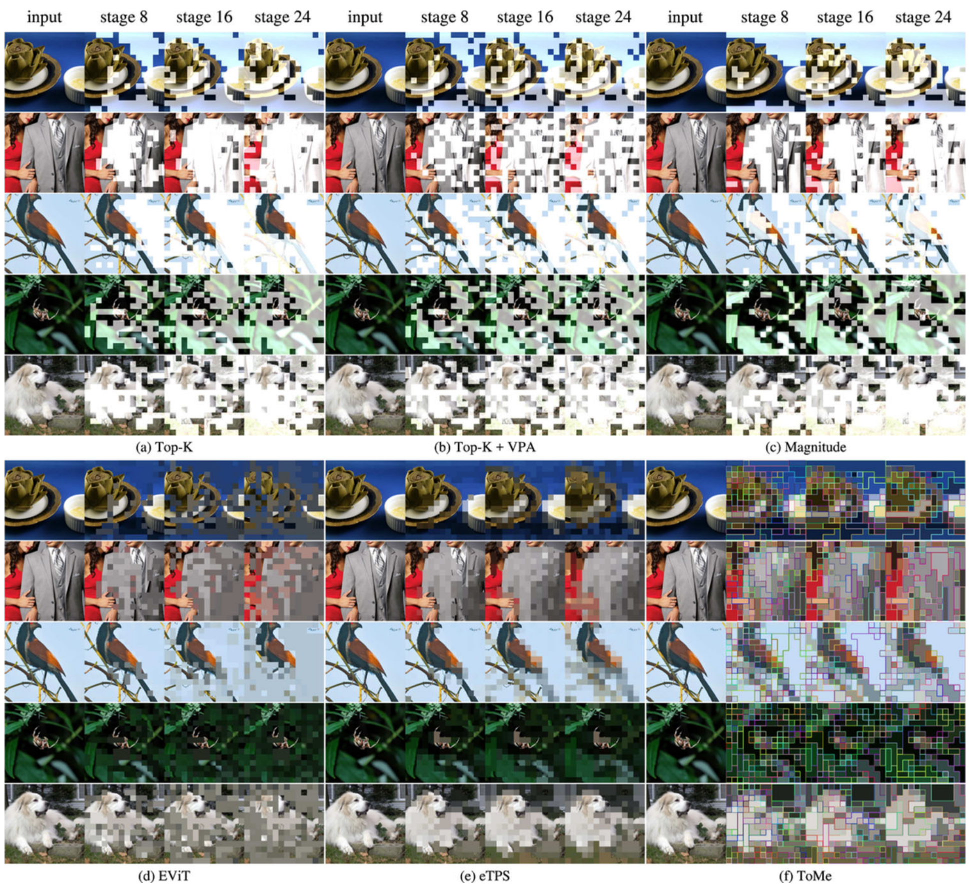

Fig. 6 Visualization of the Token Reduction for ImageNet-1k dataset: (a)–(c) are token pruning and (d)–(f) are token merging visualization. Each pruning method is applied to an off-the-shelf ViT-L with a pruning rate of 5% for each layer, without fine-tuning; since ViT-L has 24 layers of Transformer Encoders, token reduction in layers 8, 16, and 24 are visualized

for the tokens. The larger the number of tokens over which information is distributed, the more effectively VPA works. Therefore, as the results in Table 3 show, the vanilla model with a larger number of tokens has a better ability to maintain performance with VPA than the hybrid model.

5.2 Limitations

The proposed pruning rate adjustment method refers to training loss and adjusts the pruning rate so that the training loss converges. When applying the method Adjust, the following could be a problem.

1. If convergence of the training loss is difficult, the pruning rate adjustment could become unstable. Therefore, this proposal has low effectiveness in tasks where losses do not converge.
2. In unstable losses in the early stages of training, significant pruning rate changes are frequently made, and the model's performance is severely degraded due to inconsistent optimization goals. Therefore, effective adjustment of the pruning rate could be achieved if the downstream task is trained in advance without pruning and the losses are stabilized before the adjustment.
3. Because the pruning rate adjustment actively converges training losses, it could encourage over-fitting of the model in tasks that are prone to over-fitting.

5.3 Future work

To alleviate some of the limitations of Section 5.2, the following ideas could be useful

1. To stabilize the pruning rate adjustment, momentum term or exponential moving averages, which are also used in deep learning parameter optimization methods, could be used to avoid unstable pruning rates.
2. Separating the pruning rate search process from the training process to avoid inconsistent optimization could further improve results. In particular, determining the pruning rate by the recoverability of accuracy at each

layer after pruning, as proposed in DC-ViT [36], is worth considering.

3. The application of DeepDream [37] technique for generating synthetic images proposed in DC-ViT could be applied to suppress over-fitting on training data.

Since the automatic pruning rate adjustment proposed in this paper is a Top-K-based token reduction, it can only be applied to general ViTs and not to ViTs with hierarchical structures such as the Swin Transformer [13]. Therefore, token reduction in ViTs with hierarchical structures is a future issue.

6 Conclusion

In this paper, we propose a method for automatically adjusting the pruning rate for ViT token pruning while training and Variable Proportional Attention (VPA) to mitigate the performance degradation in Top-K. Experimental results show that the pruning method with pruning rate adjustment has superior Accuracy on the CIFAR-10 dataset compared to the pruning method with manual pruning rate setting. Furthermore, the proposed Top-K+VPA method achieves a latency reduction of 46.8% on GPU, 42.0% on GPU-TRT, 34.9% on CPU, and 20.8% on Egde-TRT on ViT-L when the pruning rate is set to 6%. In addition, when the pruning rate is increased, Top-K+VPA achieves recognition accuracy better than competitive token merging methods on ImageNet-1k.

Author Contributions All authors contributed to the study conception and design. Ryuto Ishibashi and Lin Meng performed material preparation, data collection, and analysis. Ryuto Ishibashi wrote the first draft of the manuscript, and all authors commented on previous versions. All authors read and approved the final manuscript. Lin Meng is a corresponding author.

Funding Part of this work was supported by JST Strategic International Collaborative Research Program (SICRP), Grant Number JPMJSC23A5.

Data Availability The CIFAR-10 and CIFAR100 datasets are publicly available datasets used for image classification tasks. The CIFAR-10 and CIFAR100 datasets were created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, and are available at CIFAR-10, CIFAR100, reference [22]. The Oxford 102 Flowers dataset (Flowers) is publicly available for academic and research purposes. The Oxford 102 Flowers dataset contains 8,189 images of 102 flower species native to the United Kingdom, with annotations for each species. The dataset can be accessed from the Visual Geometry Group (VGG) at the University of Oxford and is available at Flowers reference [23]. The Oxford-IIIT Pets dataset is publicly available for academic and research purposes. The dataset can be accessed from the Visual Geometry Group (VGG) at the University of Oxford and is available at Pets. The Oxford-IIIT Pets dataset contains 7,349 images of 37 pet breeds reference [24]. The Caltech-UCSD Birds-200-2011 (CUB-200) dataset is the dataset for fine-grained visual categorization task. CUB-200 contains 11,788 images of 200 subcategories belonging to birds, 5,994 for training and 5,794 for testing. The dataset can be accessed from the Caltech Vision Lab and is available at CUB-200 reference [25]. The Indoor Scene Recognition (Indoor67) contains 67 Indoor categories, and a total of 15,620 images. For evaluation, this dataset is available as a subset of the dataset with a partition of 80*67 training data and 20*67 test data at Indoor67 reference [26]. The ImageNet-1k dataset is a large-scale visual database designed for visual object recognition research. ImageNet was developed and maintained by researchers at Princeton University and Stanford University and is available at ImageNet reference [27].

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
2. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021) An image is worth 16x16 words: transformers for image recognition at scale. In: International conference on learning representations
3. Ren J, Li H, Wang A, Saho K, Meng L (2024) Radar-based gait analysis by transformer-liked network for dementia diagnosis. Biomed Signal Process Control
4. Ishibashi R, Nojiri N, Kaneko H, Kenshi S, Meng L (2023) Optimized vision transformer for dementia diagnosis using micro-doppler radar. In: 2023 IEEE international conference on systems, man, and cybernetics (SMC)
5. Zhang C, Liu H, Deng Y, Xie B, Li Y (2023) Tokenhpe: learning orientation tokens for efficient head pose estimation via transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 8897–8906
6. Liu H, Zhang C, Deng Y, Xie B, Liu T, Li Y-F (2023) Transifc: invariant cues-aware feature concentration learning for efficient fine-grained bird image classification. IEEE Trans Multimed 1–14
7. Liu H, Zhang C, Deng Y, Liu T, Zhang Z, Li Y-F (2023) Orientation cues-aware facial relationship representation for head pose estimation via transformer. IEEE Trans Image Process 32:6289–6302
8. Yu L, Xiang W (2023) X-pruner: explainable pruning for vision transformers. In: Proc. of the IEEE/CVF conference on computer vision and pattern recognition, pp 24355–24363
9. Bolya D, Fu C-Y, Dai X, Zhang P, Feichtenhofer C, Hoffman J (2023) Token merging: your ViT but faster. In: International conference on learning representations

10. Devlin J, Chang M-W, Lee K, Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding. In: North American chapter of the association for computational linguistics
11. Graham B, El-Nouby A, Touvron H, Stock P, Joulin A, Jégou H, Douze M (2021) Levit: a vision transformer in convnet's clothing for faster inference. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 12259–12269
12. Chu X, Tian Z, Zhang B, Wang X, Shen C (2023) Conditional positional encodings for vision transformers. In: The eleventh international conference on learning representations
13. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 10012–10022
14. Liang Y, Ge C, Tong Z, Song Y, Wang J, Xie P (2022) Not all patches are what you need: expediting vision transformers via token reorganizations. In: International conference on learning representations
15. Wei S, Ye T, Zhang S, Tang Y, Liang J (2023) Joint token pruning and squeezing towards more aggressive compression of vision transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2092–2101
16. Xu J, De Mello S, Liu S, Byeon W, Breuel T, Kautz J, Wang X (2022) Groupvit: semantic segmentation emerges from text supervision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 18134–18144
17. Rao Y, Zhao W, Liu B, Lu J, Zhou J, Hsieh C-J (2021) Dynamicvit: Efficient vision transformers with dynamic token sparsification. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Wortman Vaughan J (Eds) Advances in neural information processing systems, vol 34, pp 13937–13949. Curran Associates, Inc
18. Yin H, Vahdat A, Alvarez JM, Mallya A, Kautz J, Molchanov P (2022) A-vit: adaptive tokens for efficient vision transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10809–10818
19. Pan B, Panda R, Jiang Y, Wang Z, Feris R, Oliva A (2021) Ia-red^2: interpretability-aware redundancy reduction for vision transformers. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Wortman Vaughanand J (Eds) Advances in neural information processing systems, vol 34, pp 24898–24911. Curran Associates, Inc
20. Fayyaz M, Koohpayegani SA, Jafari FR, Sengupta S, Joze HR, Sommerlade E, Pirsiavash H, Gall J (2022) Adaptive token sampling for efficient vision transformers. In: European conference on computer vision, pp 396–414. Springer
21. Chen X, Liu Z, Tang H, Yi L (2023) Sparsevit: revisiting activation sparsity for efficient high-resolution vision transformer. In: Proc. of the IEEE/CVF conference on computer vision and pattern recognition, pp 2061–2070
22. Krizhevsky A, Hinton G, et al (2009) Learning multiple layers of features from tiny images
23. Nilsback M-E, Zisserman A (2006) A visual vocabulary for flower classification. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), vol 2, pp 1447–1454. IEEE
24. Parkhi OM, Vedaldi A, Zisserman A, Jawahar CV (2012) Cats and dogs. In: 2012 IEEE conference on computer vision and pattern recognition, pp 3498–3505. IEEE
25. Wah C, Branson S, Welinder P, Perona P, Belongie S (2011) The caltech-ucsd birds-200-2011 dataset
26. Quattoni A, Torralba A (2009) Recognizing indoor scenes. In: 2009 IEEE conference on computer vision and pattern recognition, pp 413–420. IEEE
27. Deng J, Berg A, Satheesh S, Su H, Khosla A, Fei-Fei L (2012) Imagenet large scale visual recognition competition 2012 (ilsvrc2012). See net. org/challenges/LSVRC 41:6
28. Steiner AP, Kolesnikov A, Zhai X, Wightman R, Uszkoreit J, Beyer L (2022) How to train your vit? data, augmentation, and regularization in vision transformers. Trans Mach Learn Res
29. Cubuk ED, Zoph B, Shlens J, Le QV (2020) Randaugment: practical automated data augmentation with a reduced search space
30. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: Computer vision–ECCV 2016: 14th european conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, pp 630–645. Springer
31. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
32. Wightman R, Touvron H, Jegou H (2021) Resnet strikes back: an improved training procedure in timm. In: NeurIPS 2021 workshop on imagenet: past, present, and future
33. Tan M, Le Q (2019) EfficientNet: rethinking model scaling for convolutional neural networks. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning, volume 97 of Proceedings of Machine Learning Research, pp 6105–6114. PMLR, 09–15
34. Ridnik T, Ben-Baruch E, Noy A, Zelnik-Manor L (2021) Imagenet-21k pretraining for the masses. In: Thirty-fifth conference on neural information processing systems datasets and benchmarks track (Round 1),
35. Ishibashi R, Ishikawa M, Meng L (2024) Norm-regularized token compression in vision transformer networks
36. Zhang H, Zhou Y, Wang G-H, Wu J (2024) Dense vision transformer compression with few samples. In: 2024 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 15825–15834
37. Mordvintsev A, Olah C, Tyka M (2015) Inceptionism: going deeper into neural networks

Ryuto Ishibashi received his bachelor's degree from the Department of Electronic and Computer Engineering, College of Science and Engineering, Ritsumeikan University, Kusatsu, Japan, in 2023. He is currently a master's student in the Department of Electronic Systems at the Graduate School of Science and Engineering, Ritsumeikan University. His research interests include deep learning, computer vision, and lightweight AI models. He is a student member of IEEE.

Lin Meng is a Professor at the College of Science and Engineering and head of the Intelligent High-performance Computing Lab., at Ritsumeikan University (RU), Japan. He received his Ph.D. from the Graduate School of SE at RU in 2012. He was a Research Associate, Assistant Professor, and lecturer at the Dept. of ECE, RU. In 2015, he was a visiting scholar in the Dept. of CSE, University of Minnesota at Twin Cities, USA. His research interests include Computer Architecture, Parallel Processing, Intelligence High-Performance Computing (IHPC), FPGA-based Accelerator Design, the Internet of Things (IoT), and Artificial Intelligence (AI). He is a senior member of IEEE and a member of ACM, IPSJ, IEICE, and IEE.