DiffRate : Differentiable Compression Rate for Efficient Vision Transformers

Mengzhao Chen[1,2†‡], Wenqi Shao[2‡], Peng Xu[2,3], Mingbao Lin[4], Kaipeng Zhang[2], Fei Chao[1], Rongrong Ji[1*], Yu Qiao[2], Ping Luo[2,3]
[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, School of Informatics, Xiamen University
[2]OpenGVLab, Shanghai AI Laboratory [3]The University of Hong Kong [4]Tencent Holdings Ltd

Abstract

*Token compression aims to speed up large-scale vision transformers (*e.g. *ViTs) by pruning (dropping) or merging tokens. It is an important but challenging task. Although recent advanced approaches achieved great success, they need to carefully handcraft a compression rate (*i.e. *number of tokens to remove), which is tedious and leads to sub-optimal performance. To tackle this problem, we propose* ***Diff****erentiable Compression* ***Rate*** *(DiffRate), a novel token compression method that has several appealing properties prior arts do not have. First, DiffRate enables propagating the loss function's gradient onto the compression ratio, which is considered as a non-differentiable hyperparameter in previous work. In this case, different layers can automatically learn different compression rates layer-wisely without extra overhead. Second, token pruning and merging can be naturally performed simultaneously in DiffRate, while they were isolated in previous works. Third, extensive experiments demonstrate that DiffRate achieves state-of-the-art performance. For example, by applying the learned layer-wise compression rates to an off-the-shelf ViT-H (MAE) model, we achieve a 40% FLOPs reduction and a 1.5× throughput improvement, with a minor accuracy drop of 0.16% on ImageNet without fine-tuning, even outperforming previous methods with fine-tuning. Codes and models are available at* https://github.com/OpenGVLab/DiffRate.

1. Introduction

Vision Transformer (ViT) [7] has rapidly developed and achieved state-of-the-art performance in various vision tasks such as image classification[26], object detection [45], and semantic segmentation [36, 12, 15, 16]. Due to the flexibility in handling various input formats, ViT has also been widely applied to self-supervised learning [13] and other modalities [9, 37]. Despite the remarkable success, ViTs suffer from the intensive computational complexity that increases quadratically with the token length in the self-attention layer, presenting a challenge for practical applications. Therefore, it is crucial to improve the efficiency of ViTs in order to make them more widespread.

In pursuit of efficient ViTs, various network compression techniques such as weight pruning [39, 34], quantization [27, 42], distillation [17, 35] and so on have been investigated. Among them, token compression[29, 1, 23]

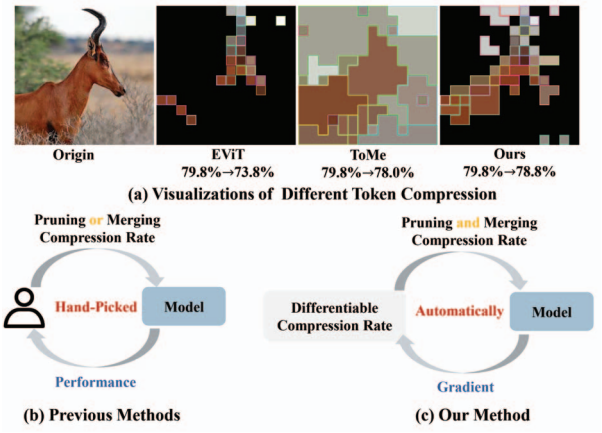


Figure 1: **Comparison of different token compression methods** including token pruning EViT [23], token merging ToMe [1] and our method. Pruned tokens are represented by black and non-border, while merged tokens are represented by patches with the same inner and border color. (**a**) shows that our method achieves better top-1 accuracy on ImageNet with FLOPs of 2.3G when compressing pre-trained Deit-S [32] without fine-tuning. (**b**) and (**c**) show that previous methods typically focus on either pruning or merging tokens using hand-picked compression rate with the guidance of performance. But our method leverages both approaches simultaneously to achieve more effective compression using the differentiable compression rate with gradient optimization.

*Corresponding authors: Rongrong Ji (rrji@xmu.edu.cn)
† This work was done during his internship at Shanghai AI Laboratory.
‡ Equal Contribution

has emerged as a promising approach to reduce redundancy in ViT for several appealing properties. First, token compression can be applied without any modifications to the network structure by exploiting the input-agnostic nature of transformers. Second, token compression is orthogonal to previous compression methods, making it a complementary approach to existing techniques [34, 4].

Existing token compression approaches typically include token pruning [29, 23, 20] and token merging [1]. As shown in Fig. 1(a), token pruning preserves informative tokens by measuring the importance of tokens with a defined metric. It can easily identify irrelevant background tokens with low importance. On the other hand, token merging compresses tokens by merging them with a large semantic similarity, which can not only discard some background tokens but also merge less informative foreground tokens. However, both pruning and merging handcraft a compression rate (*i.e*. the ratio of removed tokens and total tokens) for each transformer layer as shown in Fig. 1(b), which has two drawbacks. First, since model complexity metrics such as FLOPs are related to compression rates in each layer, it is tedious for practitioners to set layer-wise compression rates in order to meet the complexity constraints while retaining the performance of ViTs as much as possible. Second, informative foreground tokens are prone to being discarded with a hand-picked compression rate, which results in performance degradation. As shown in Fig. 1(a), when compressing tokens at fixed rates, token pruning such as EViT [23] removes most of the informative foreground tokens while token merging [1] also merges many important foreground tokens into a single token, leading to a sudden drop of top-1 accuracy on ImageNet.

To tackle the above issues, this work proposes a unified token compression framework called Differentiable Compression Rate (DiffRate), where both pruning and merging compression rates are determined in a differentiable manner. To achieve this goal, we propose a novel method, namely Differentiable Discrete Proxy (DDP) module. In DDP, a token sorting procedure is first performed to identify important tokens with a token importance metric. Then, a re-parameterization trick enables us to optimally select top-K important tokens with gradient back-propagation. In this way, all input images would have the top-K important tokens preserved, making it possible for parallel batch computation. Notably, the optimization process of DiffRate is highly efficient and can converge within 3 epochs (*i.e*. 2.7 GPU hours for ViT-B).

Thanks to the inclusion of differentiable compression rates, DiffRate can leverage the benefits of token pruning and merging by seamlessly integrating both techniques into a forward pass. This is possible because both token pruning and merging are capable of determining the optimal set of tokens to preserve. As shown in Fig. 1(a), DiffRate can prune most irrelevant background tokens and preserve detailed foreground information, leading to a good trade-off between efficiency and performance. With the learned compression rate, DiffRate achieves state-of-the-art performance in compressing various ViTs. For example, DiffRate can compress an off-the-shelf ViT-H model pre-trained by MAE [13] with 40% FLOPs reduction and 50% throughput improvement with only 0.16% accuracy drop, outperforming previous methods that require tuning the network parameter.

Our contributions are summarized as follows:

- We develop a unified token compression framework, Differentiable Compression Rate (DiffRate), that includes both token pruning and merging, and formulate token compression as an optimization problem.
- DiffRate employs a Differentiable Discrete Proxy which consists of a token sorting procedure and a re-parameterization trick to determine the optimal compression rate under different computation cost constraints. To our knowledge, it is the first study to explore differentiable compression rate optimization in token compression.
- Through extensive experiments, we demonstrate that DiffRate outperforms previous methods and achieves state-of-the-art performance on the off-the-shelf models. We hope that DiffRate can advance the field of token compression and improve the practical application of Vision Transformers (ViTs).

2. Related Work

Token Compression Several recent studies have attempt compress redundancy token according token pruning [23, 29, 8, 40, 31, 21, 20, 38, 24, 34] and token merging [1, 43, 30, 28]. However, most of these methods focus on designing metrics to distinguish redundant tokens, while ignoring the token compression schedule in each block. Some methods, such as VTC-LFC [34] and ViT-Slim [4] combine token prune with weight prune and determine the number of prune tokens using threshold-based approaches, which is also highly influenced by the hand-picked hyper-parameters. In contrast, our proposed method can learn the token compression schedule in a differentiable form. Furthermore, we consider both token merging and token pruning in the token compression process, resulting in a better trade-off between speed and accuracy.

Differentiable Neural Architecture Search. There are also many works [25, 3, 11] attempted to search for neural architecture in a differentiable manner. For example, DARTS [25] and ProxylessNAS [3] learn the probabilities of each candidate operation and select the operation with the highest probability as the final architecture. DMCP [11]

is the approach most similar to proposed DiffRate, as it searches for the channel number of convolution in each layer. However, the differentiable method used in channel pruning cannot be directly applied to token compression. Firstly, the definition of search space is different. The search space for channel pruning is the channel, while for token compression, it is the token. Channels in the same position represent the same feature, while tokens are position-agnostic. Secondly, the output channel number in each layer is independent to other layers while the token must be pruned once it is discarded in previous layers. As of now, no approach has been proposed for differentiable token compression rate.

3. Differentiable Compression Rate

In this section, we first briefly introduce a transformer block and existing compression approaches, and then present our Differentiable Compression Rate (DiffRate) to build a unified token compression technique.

Transformer Block. Token compression in ViTs operates in each transformer block. Given the input token of the l-th block $\mathbf{X}^l \in \mathbb{R}^{N\times D}$ where N and D are the token length and token size, respectively, the forward propagation of the transformer block is expressed as follows:

$$\hat{\mathbf{X}}^l = \mathbf{X}^l + \mathrm{Attention}(\mathbf{X}^l), \mathbf{X}^{l+1} = \hat{\mathbf{X}}^l + \mathrm{MLP}(\hat{\mathbf{X}}^l), \quad (1)$$

where $l \in [L]$ and L is the network depth. Moreover, $\mathrm{Attention}$ and MLP represent the self-attention and the MLP modules in the transformer block, respectively. In Eqn. (1), $\hat{\mathbf{X}}^l$ is the output token of Attention.

Token Pruning and Merging. As shown in Fig. 2, existing token compression methods usually remove redundant tokens from $\hat{\mathbf{X}}^l$ by token pruning or merging, as given by

$$\hat{\mathbf{X}}^l_p \leftarrow f_p(\hat{\mathbf{X}}^l, \alpha^l_p) \text{ or } \hat{\mathbf{X}}^l_m \leftarrow f_m(\hat{\mathbf{X}}^l, \alpha^l_m), \quad (2)$$

where f_p, f_m are pruning and merging operations,α^l_p, α^l_m are their compression rates, and $\hat{\mathbf{X}}^l_p \in \mathbb{R}^{N^l_p\times D}, \hat{\mathbf{X}}^l_m \in \mathbb{R}^{N^l_m\times D}$ are their outputs which are then fed into MLP in Eqn. (1). Hence, the pruning and merging compression rate for each block is defined as $\alpha^l_p = (N - N^l_p)/N$ and $\alpha^l_m = (N - N^l_m)/N$, respectively. For example, EViT [23] preserves the important tokens while fusing unimportant tokens between Attention and MLP under the guidance of an importance metric. ToMe [1] merges similar tokens of $\hat{\mathbf{X}}^l$ in both foreground and background. Note that DynamicViT [29] prunes tokens after MLP, but we find that it also works well when it operates after Attention. Although these approaches achieved great success, they need to carefully handcraft a compression rate block-wisely, which is tedious and leads to sub-optimal performance as shown in Fig. 1(a).

Unified Formulation of DiffRate. To mitigate this problem, we propose Differentiable Compression Rate (DiffRate), a unified token compression method, to search compression rates optimally. Given a pre-trained model $\mathbf{W}^*$, the objective of token compression is to minimize the classification loss $\mathcal{L}_{cls}$ on a training dataset $(\mathbf{X}, \mathbf{Y})$ with target FLOPs T. This can be formulated as an optimization problem as follows:

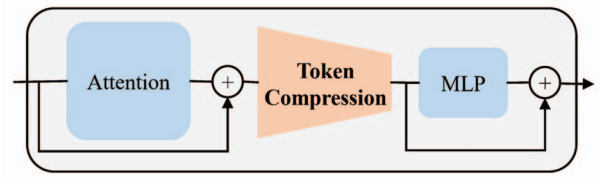


Figure 2: **Token compression** in a transformer block.

$$\boldsymbol{\alpha}^*_p, \boldsymbol{\alpha}^*_m = \arg\min_{\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m} \mathcal{L}_{cls}(\mathbf{W}^*(\mathbf{X}), \mathbf{Y} | \boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m), \quad (3)$$

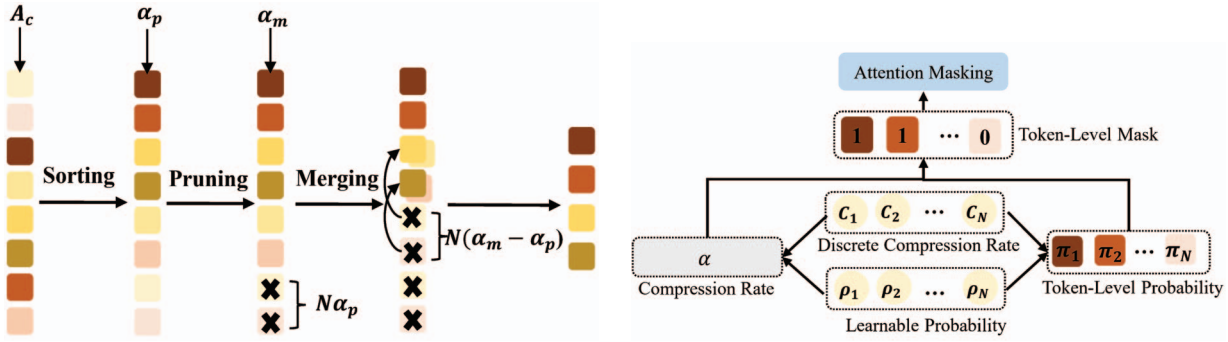
$$\text{s.t. } \mathcal{F}(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m) \le T, 0 \le \alpha^l_p, \alpha^l_m \le 1, \quad (4)$$

$$\hat{\mathbf{X}}^l = f_c(\hat{\mathbf{X}}^l, \alpha^l_p, \alpha^l_m), l \in [L] \quad (5)$$

where $\boldsymbol{\alpha}_p = \{\alpha^l_p\}^L_{l=1}$ and $\boldsymbol{\alpha}_m = \{\alpha^l_m\}^L_{l=1}$ represent pruning and merging compression rates across all blocks, respectively. Moreover, $\mathcal{F}(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m)$ denotes the corresponding FLOPs, which can be expressed as a differentiable way of the compression rates. Eqn. (5) shows that DiffRate compresses $\hat{\mathbf{X}}^l$ with operation f_c and compression rates α^l_p and α^l_m in each transformer block. Finally, $\boldsymbol{\alpha}^*_p$ and $\boldsymbol{\alpha}^*_m$ are obtained by differentiably learning in DiffRate.

With the unified formulation of token compression, DiffRate is capable enough to express various compression methods. In detail, when $f_c = f_p, \alpha^l_m = 0$, DiffRate represents token pruning with differentiable pruning compression rate α^l_p. when $f_c = f_m, \alpha^l_p = 0$, DiffRate turn into differentiable token merging. In this work, we set $f_c = f_m \circ f_p$, meaning that tokens are first pruned and then merged. In this case, DiffRate seamlessly integrates token pruning and token merging through differentiable compression rates.

However, it is challenging to solve the optimization problem in Eqn. (3-5) with gradient-based methods for ViTs as compression rates are not differentiable. Directly learning 0-1 masks for tokens as did in channel pruning [19] is infeasible because each image may drop different numbers of tokens. This makes it hard to parallelize the computation. For example, DynamicViT [29] and SPViT [20] maintain a mask vector for each input image, but they still need to manually design compression rates to ensure that all images preserve the same number of tokens. The following section introduces a novel technique for the differentiable search of compression rates.



(a) Token Sorting (b) Compression Rate Re-parameterization

Figure 3: **Pipeline of Differentiable Discrete Proxy**. (a) Token Sorting: the input N tokens are sorted based on the importance metric class attention A_c. With pruning rate α_p and merging rate α_m, we first prune $N\alpha^p$ least important tokens, then merge $N(\alpha^m - \alpha^p)$ unimportant tokens with similar ones among the remaining tokens. (b) Compression Rate Re-parameterization: the approach translate compression rate α into the combination with discrete rate $\mathbf{C}$ and learnable probability ρ. The top part is attention masking [29], which simulates token dropping by mask during training.

4. Differentiable Discrete Proxy

To make compression rates differentiable, our main idea is to preserve the top-K important tokens for all images, which can not only allow for parallel batch computation but also retain the performance of the original ViTs as much as possible. To achieve this, we introduce a novel method called the Differentiable Discrete Proxy (DDP), which comprises two critical components: a token sorting procedure to identify important tokens with a token importance metric and a re-parameterization trick to optimally select top-K important tokens with gradient back-propagation. The overall pipeline of DDP is illustrated in Fig. 3.

4.1. Token Sorting

Token Importance Metric. To find top-K importance tokens, we sort tokens by token importance metric, which has been well established in the literature. Here, we employ the class attention $\mathbf{A}_c \in \mathbb{R}^{1\times N}$ as the importance metric following EViT [23]. The interaction between class attention and image tokes can be written by:

$$\mathbf{A}_c = \text{Softmax}\big(\mathbf{q}_c\mathbf{K}^T/\sqrt{D}\big), \text{ and } \mathbf{X}_c = \mathbf{A}_c\mathbf{V}, \quad (6)$$

where $\mathbf{q}_c \in \mathbb{R}^{1\times D}$, $\mathbf{K} \in \mathbb{R}^{N\times D}$, $\mathbf{V} \in \mathbb{R}^{N\times D}$ and $\mathbf{X}_c \in \mathbb{R}^{1\times D}$ denote the query vector of class token, the key matrix, the value matrix, and the class token of the self-attention layer. From Eqn. (6), the class attention $\mathbf{A}_c$ measures how much each image token contributes to the class token. Higher class attention indicates a more significant influence of the corresponding image token on the final output, implying greater importance [5, 23]. We also investigate other importance metrics in the ablation study as shown in Table 4a.

Pruning and Merging in DiffRate. With the token importance established in Eqn. (6), it is natural to remove tokens with low importance, such as those representing semantically irrelevant backgrounds by following principles in token pruning [23, 29]. As shown in Fig. 3a, we prune $N\alpha_p$ unimportant tokens in the l-th transformer block. Notely, we drop the superscript l of compression rate to simplify the notation. After that, we use cosine similarity to measure the similarity between $N(\alpha_m - \alpha_p)$ unimportant tokens and the remaining tokens. For similar token pairs, we generate a new token by directly average them. Through the above sorting-pruning-merging pipeline, the number of tokens to be pruned and merged in each block is optimally determined with learnable compression rate in our DiffRate. Hence, DiffRate can seamlessly integrate token pruning and merging.

4.2. Compression Rate Re-parameterization

DDP uses a re-parameterization trick to make pruning and merging compression rates differentiable. We simplify the notation by using a single variable α to represent both compression rates.

Re-parameterization with Discrete Rates. In essential, making compression rate differentiable is to determine how many tokens should be discarded with optimality guarantee. To tackle this problem, we re-parameterize the compression rate as a learnable combination of multiple candidate compression rates. Specifically, we introduce a discrete compression rate set, denoted as $\mathbf{C} = \{C_1, C_2, ..., C_N\}$, where $C_k = \frac{k-1}{N}$ represents the top $(k-1)$ least important tokens should be removed. By assigning learnable probabilities ρ_k to each candidate compression rate C_k with $\sum_{k=1}^{N} \rho_k = 1$,

the compression rate can be written as

$$\alpha = \sum_{k=1}^{N} C_k \rho_k. \tag{7}$$

By using discrete candidate rates as the proxy, the optimization problem of learning compression rates can be translated into the problem of learning the probabilities ρ_k.

Token-Level Mask. As shown in Fig. 3b, with C_k and ρ_k, the probability that the k-th important token is compressed can be calculated as

$$\pi_1 = 0, \pi_k = \rho_{N+2-k} + \cdots + \rho_{N-1} + \rho_N, k \geq 2 \tag{8}$$

where $\pi_1 = 0$ indicates that the most important token is always retained. From Eqn. (8), it is easy to see that $\pi_k \leq \pi_{k+1}$. Therefore, our DiffRate with DDP aligns with the fact that less important tokens should have a larger compression probability. To make the training and inference consistent, we convert π_k into a 0-1 mask as given by,

$$m_k = \begin{cases} 0, \pi_k \geq \alpha, \\ 1, \pi_k < \alpha, \end{cases} \tag{9}$$

where $m_k = 1$ indicates that the k-th token is preserved, and vice versa.

In each vision transformer block, we instantiate two independent re-parameterization modules to learn both pruning and merging compression rates. Thus, it generates two token-level masks, namely the pruning mask and the merging mask, denoted m_k^p and m_k^m for each token, respectively. Note that the token removed in last block must also be compressed in this block. Hence, the final mask is defined as

$$m_k = m_k \cdot m_k^p \cdot m_k^m, \tag{10}$$

where m_k on the right-hand side is the mask for the k-th token in the last block.

Attention Masking. To preserve the gradient back-propagation chain, we convert token dropping into attention masking with mask m_k in Eqn. (10) following DynamicViT [29]. To achieve this, we construct an attention mask $\mathbf{M}$ with the same dimensions as the attention map for each self-attention operation:

$$M_{i,j} = \begin{cases} 1, i = j, \\ m_j, i \neq j. \end{cases} \tag{11}$$

The attention mask prevents the interaction between all compressed tokens and the other tokens, except itself. We then use this mask to modified the $\mathrm{Softmax}$ operation in the next self-attention module:

$$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D}}, \hat{S}_{i,j} = \frac{\exp(S_{i,j}) M_{i,j}}{\sum_{k=1}^{N} \exp(\mathrm{S_{i,k}}) M_{i,k}}, \tag{12}$$

Algorithm 1 Differentiable Compression Rate.

Input: training dataset $(\mathbf{X}, \mathbf{Y})$, pretrained model weight $\mathbf{W}^*$, target FLOPs T, DDP with discrete compression rate $\{C_k\}_{k=1}^N$ and learnable probabilities $\{\rho_k\}_{k=1}^N$.
Output: block-wise pruning compression rate $\boldsymbol{\alpha}_p = \{\alpha_p^l\}_{l=1}^L$ and merging compression rate $\boldsymbol{\alpha}_m = \{\alpha_m^l\}_{l=1}^L$.

1: **for** $(\mathbf{x}, \mathbf{y})$ in $(\mathbf{X}, \mathbf{Y})$ **do**
2: calculate $\boldsymbol{\alpha}_p$ and $\boldsymbol{\alpha}_m$ with $\{\rho_k\}_{k=1}^N$ by Eqn. (7).
3: calculate pruning mask $\{m_k^p\}_{k=1}^N$ by Eqn. (9).
4: calculate merging mask $\{m_k^m\}_{k=1}^N$ by Eqn. (9).
5: sorting $\rightarrow$ pruning $\rightarrow$ merging in Sec. 4.1
6: attention masking with Eqn. (11-12)
7: calculate classification loss: $\mathcal{L}_{cls}(\mathbf{W}^*(\mathbf{x}), \mathbf{y})$
8: calculate FLOPs loss: $L_f = (\mathcal{F}(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m) - T)^2$
9: calculate optimization objective: $\mathcal{L} = \mathcal{L}_{cls} + \lambda_f \mathcal{L}_f$
10: backward $\mathcal{L}$ to ρ_k with Eqn. (14)
11: update ρ_k by gradient
12: **end for**
13: **return** $\boldsymbol{\alpha}_p$ and $\boldsymbol{\alpha}_m$

where $\mathbf{Q} \in \mathbb{R}^{N \times D}$ is the query matrix, $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the original attention map before $\mathrm{SoftMax}$, and $\hat{S}_{i,j}$ is actually used to update tokens. Eqn. (11-12) enables propagation the loss function's gradient onto the mask m.

5. Training Objective

We solve the optimization problem in Eqn. (3-5) by minimizing the total loss as given by

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda_f \mathcal{L}_f(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m), \tag{13}$$

where $\mathcal{L}_f = (\mathcal{F}(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m) - T)^2$ is the loss to constraint the FLOPs. The hyper-parameter λ_f balances the two loss terms, and we set it to 5 by default in our experiments.

During network back-propagation, we utilize the straight-through-estimator (STE) [14] to calculate the gradient of Eqn. (11). Hence, we can calculate the gradient of $\mathcal{L}$ with respect to ρ_k using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \rho_k} = \sum_{j=1}^{N} \frac{\partial \mathcal{L}}{\partial m_j} \frac{\partial m_j}{\partial \pi_j} \frac{\partial \pi_j}{\partial \rho_k} \approx \sum_{j=1}^{N} \frac{\partial \mathcal{L}}{\partial m_j} \frac{\partial \pi_j}{\partial \rho_k}. \tag{14}$$

Since ρ_k is differentiable through Eqn. (14), the compression rate α can be optimized with gradient back-propagation by Eqn. (7).

Overall Algorithm. The overall training algorithm of DiffRate is illustrated in Algorithm 1. It consists of three steps: forward model with ρ_k (Lines 2-6), calculating optimization objective (Lines 7-9), backward propagation and ρ_k update in DDP (Lines 10-11). The DiffRate algorithm finds the optimal compression rate by updating ρ_k in a differentiable form, and the resulting compression rate can be directly applied to off-the-shelf models.

Extension to Other Complexity Metrics. Our DiffRate model offers flexible differentiable compression rates that can be supervised using various computational complexity metrics, such as FLOPs and latency. To investigate its potential, we employed Gemmini [10], a framework for generating deep learning accelerators that can produce a diverse set of realistic accelerators based on a flexible architecture template. Using Gemmini, we conducted a co-search of the design space for compression ratios $\boldsymbol{\alpha}_p$ and $\boldsymbol{\alpha}_m$, as well as accelerator parameters $\boldsymbol{\beta}$ simultaneously. The loss function in Eqn. (13) becomes

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{cls} + \lambda_f \mathcal{L}_f(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m) \\ &+ \lambda_{la}\mathcal{L}_{la}(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m, \boldsymbol{\beta}) + \lambda_{pw}\mathcal{L}_{pw}(\boldsymbol{\alpha}_p, \boldsymbol{\alpha}_m, \boldsymbol{\beta}),\end{aligned} \tag{15}$$

where $\mathcal{L}_{la}$ and $\mathcal{L}_{pw}$ are loss functions to constraint latency and power consumption in Gemmini accelerator, respectively. λ_{la} and λ_{pw} are their strengths. By minimizing $\mathcal{L}$, DiffRate can further learn optimal $\boldsymbol{\alpha}_p^*, \boldsymbol{\alpha}_m^*$ satisfying various resource constraints, and the optimum $\boldsymbol{\beta}^*$ can be implemented in FPGA board. The details are in Appendix B.

6. Experiments

This section presents extensive experiments to verify our proposed DiffRate. Sec. 6.1 provides training details. Sec. 6.2 compares DiffRate with previous compression techniques. The ablation study and visualization are presented in Sec. 6.3 and Sec. 6.4, repsectively.

6.1. Implementation Details

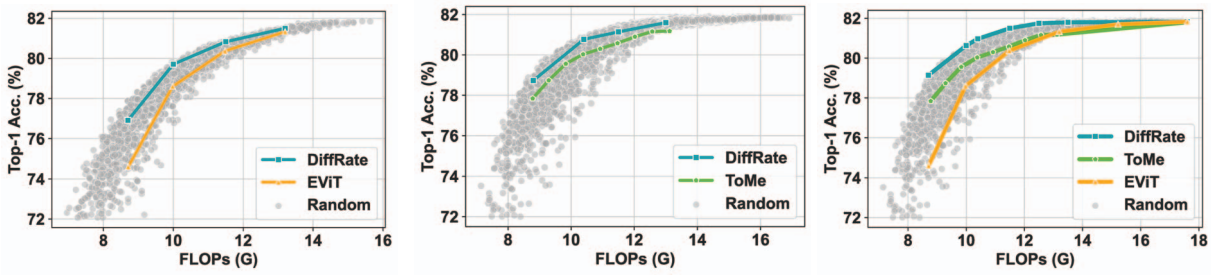
In this section, we conduct a series of experiments on ImageNet-1k [6] using DeiT [32], MAE [13], and LV-ViT [18]. We initialize the backbone models with pre-trained models and fix them to train the learnable probabilities in DDP with the objective function in Eqn. (13). The DDP is trained for 3 epochs using a learning rate of $1e^{-2}$. Once the compression rate of DDP is determined, it can be directly applied to off-the-shelf models. We also fine-tune the model optionally for 30 epochs using a learning rate of $1e^{-5}$ after determining the compression rate, denoted by the superscript "†" in the table for differentiation. All throughput measurements are taken during inference on an A100 GPU with a batch size of 1024 and fp16. The other experimental setups follow most of the training techniques used in DeiT [32], and additional details can be found in Appendix A. Note that our DiffRate is highlighted in the tables in gray, and **bold** denotes the best results.

6.2. Comparison with state-of-the-art

Compression Schedule. In Fig. 4, we compare the compression rate obtained by DiffRate with three other token pruning schedules, including EViT [43], ToMe [1], and randomly sampled schedules. We evaluate the FLOPs and accuracy on the ImageNet-1k validation dataset using an off-the-shelf ViT-B (DeiT) and investigate three token compression options: only pruning, only merging, and a combination of pruning and merging, as depicted in Sec. 4.1. We can observe that DiffRate performed almost optimally, regardless of the token compression setting and FLOPs constraint. Additionally, DiffRate's advantage was more prominent at lower FLOPs constraints, indicating its ability to provide more appropriate compression rates at larger solution space. DiffRate also benefited from unified token compression, indicating that it can preserve more information by combining

Table 1: **Token compression on the off-the-shelf models.** We directly apply EViT [23], ToMe [1], and the proposed DiffRate to off-the-shelf models, *i.e.* without updating the network parameters of pre-trained models.

Model	Method	FLOPs	imgs/s	Acc.
ViT-S (DeiT)	Baseline [32]	4.6	5039	79.82
	EViT [23]	2.3	8950	73.83
	ToMe [1]	2.3	8874	77.99
	DiffRate	**2.3**	8901	**78.76**
	EViT [23]	3.0	6807	78.50
	ToMe [1]	2.9	6712	78.89
	DiffRate	**2.9**	6744	**79.58**
ViT-B (DeiT)	Baseline [32]	17.6	2130	81.83
	EViT [23]	8.7	4230	74.61
	ToMe [1]	8.8	4023	77.84
	DiffRate	**8.7**	4124	**78.98**
	EViT [23]	11.5	2886	80.37
	ToMe [1]	11.5	2834	80.58
	DiffRate	**11.5**	2865	**81.50**
ViT-B (MAE)	Baseline [13]	17.6	2130	83.72
	EViT [23]	8.7	4230	75.15
	ToMe [1]	8.8	4023	78.86
	DiffRate	**8.7**	4150	**79.96**
	EViT [23]	11.5	2886	82.01
	ToMe [1]	11.5	2834	82.32
	DiffRate	**11.5**	2865	**82.91**
ViT-L (MAE)	Baseline [13]	61.6	758	85.95
	EViT [23]	29.7	1672	81.52
	ToMe [1]	31.0	1550	84.24
	DiffRate	**31.0**	1580	**84.65**
	EViT [23]	39.6	1089	85.06
	ToMe [1]	42.3	1033	85.41
	DiffRate	**42.3**	1045	**85.56**
ViT-H (MAE)	Baseline [13]	167.4	299	86.88
	ToMe [1]	92.9	500	86.01
	EViT [23]	99.1	512	85.54
	DiffRate	**93.2**	504	**86.40**
	ToMe [1]	103.4	442	86.29
	EViT [23]	112.9	432	86.32
	DiffRate	**103.4**	450	**86.72**
LV-ViT-S	Baseline [18]	6.6	3630	83.30
	EViT [23]	3.9	5077	79.77
	DiffRate	**3.9**	5021	**82.56**



(a) Pruning (b) Merging (c) Pruning and Merging

Figure 4: **Token Compression Schedule Comparison.** We test the proposed DiffRate on ViT-B (DeiT) with three token compression options: (a) Only token pruning like EViT [43], (b) Only token merging like ToMe [1],(c) Pruning and Merging as depicted in Sec. 4.1. The token compression schedules searched through DiffRate outperform the constant schedules used in ToMe [1] and EViT [23]. Moreover, the performance of our method is close to optimal when compared to $10,000$ randomly sampled schedules.

the merits of token merging and pruning.

Off-the-shelf. We compare the proposed DiffRate with EViT [43] and ToMe [1] in the "off-the-shelf" setting, indicating direct use of the pre-trained model without updating the network parameters. Notably, the compression rate achieved by DiffRate can be applied to pre-trained models without any additional computation. As shown in Table 1, DiffRate consistently outperforms both EViT [23] and ToMe [1] across various models. Specifically, DiffRate outperforms ToMe [1] by 0.14% to 1.14%, and outperforms EViT [23] by 0.39% to 4.93%. These results demonstrate the strong potential of DiffRate as an effective post-training token compression method. More results at other FLOPs constraints can be found in Appendix C.

With Training. In Table 2, we compare the proposed DiffRate with several methods that require training the model, indicating fine-tuning with pre-trained models or training from scratch. The evaluated methods include EViT [1] and ToMe [1], which train models from scratch, and ATS [8], DynamicViT [29], SP-ViT [20], which fine-tune pre-trained models for 30 epochs. To ensure a fair comparison, for DiffRate, we also fine-tune the pre-trained models for 30 epochs with the searched compression rate. We can observe that DiffRate still maintains a performance advantage compared to methods that require training. Specifically, ATS achieves a top-1 accuracy of 79.70% in DeiT-S, close to our 79.83%. However, it is important to note that ATS is an input-adaptive token pruning method that cannot be applied to batch inference, while DiffRate does not suffer this problem. Moreover, we find that DiffRate utilized with an off-the-shelf model achieves comparable or superior performance to methods that require training. For instance, DiffRate attains 79.58% on the off-the-shelf DeiT-S [32], while EViT and ToMe, after training, only achieve 79.50% and 79.49%, respectively. Similar results are also observed in ViT-B (DeiT) and ViT-B/L/H (MAE).

Table 2: **Token compression with training.** † indicates fine-tuning the model with searched compression rate for 30 epochs.

Model	Method	FLOPs	imgs/s	Acc.
ViT-S (DeiT)	Baseline [32]	4.6	5039	79.82
	DynamicViT [29]	2.9	6527	79.30
	Evo-ViT [38]	3.0	6679	79.40
	EViT [23]	3.0	6807	79.50
	ToMe [1]	2.9	6712	79.49
	ATS [8]	2.9	-	79.70
	SPViT [20]	2.6	-	79.34
	DiffRate	**2.9**	6744	**79.58**
	DiffRate†	**2.9**	6744	**79.83**
ViT-B (DeiT)	Baseline [32]	17.6	2130	81.83
	EViT [23]	11.5	2886	81.30
	ToMe [1]	11.5	2834	81.41
	DiffRate	**11.5**	2865	**81.50**
	DiffRate†	**11.5**	2865	**81.71**
ViT-B (MAE)	Baseline [13]	17.6	2130	83.72
	ToMe [1]	11.5	2834	82.94
	DiffRate	**11.5**	2865	**82.91**
	DiffRate†	**11.5**	2865	**83.25**
ViT-L (MAE)	Baseline [13]	61.6	758	85.95
	ToMe [1]	42.3	1033	85.59
	DiffRate	**42.3**	1045	**85.56**
	DiffRate†	**42.3**	1045	**85.71**
ViT-H (MAE)	Baseline [13]	167.4	299	86.88
	ToMe [1]	103.4	442	86.51
	DiffRate	**103.4**	450	**86.72**

Multiple Complexity Constraints. In addition, we supervise DiffRate with three computational complexity constraints: FLOPs, latency, and power, as detailed in Eqn. (15). As shown in Table 3, integrating multiple complexity constraints enhances the trade-off between performance and complexity. Specifically, leveraging multiple

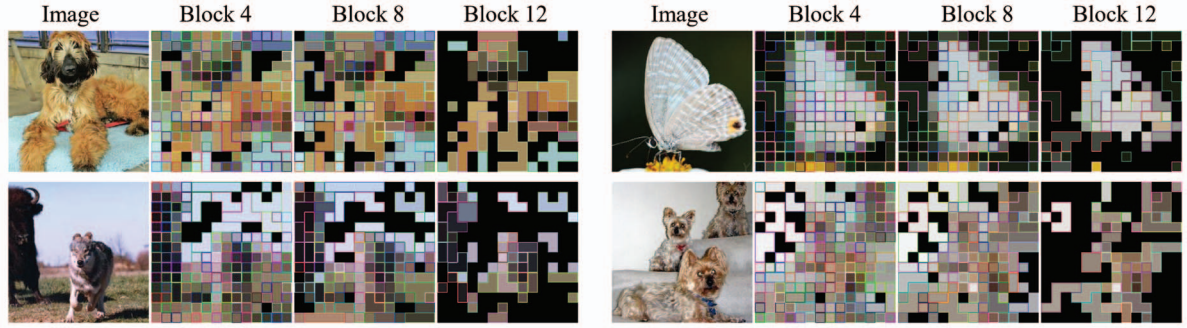


Figure 5: **Image Visualizations.** Results of proposed DiffRate on ImageNet-1k validation using a pre-trained DeiT-B model, only 34 tokens left in block 12. Merged tokens are represented by patches with the same inner and border color, while pruned tokens are represented by black. The visualizations show that DiffRate can gradually prune the redundant token in the background and merge less-discriminative tokens in the foreground.

Table 3: **Results under multiple complexity constraints.** on ViT-S (DeiT). DiffRate indicates single FLOPs constraint as Eqn. (13), DiffRate-M indicates multiple complexity constraint as Eqn. (15).

Method	FLOPs(G)	Latency(ms)	Power(mW)	Acc.
Baseline	4.6	68.1	156	79.82
EViT	3.0	40.4	99	79.50
DiffRate	2.9	40.1	98	**79.83**
DiffRate-M	**2.9**	**37.6**	**90**	79.80

complexity constraints leads to a remarkable reduction of 2.5ms in latency and 8mW in power consumption compared to a single FLOPs constraint, with only a negligible decline in accuracy of a mere 0.03%.

6.3. Ablation Study

In this section, we perform DiffRate with variants to investigate the effectiveness of our proposed method. As the experiment shown in Table 4a, we compare the influence of several token sorting metrics, including randomly generated rank, class attention (A_c), image attention (A_i) [23], and the product of class attention and the value matrix's norm ($A_c \cdot |V|$) [8]. It can be observed that A_c and $A_c \cdot |V|$ exhibit similar performance, and we choose A_c as our default setting since it does not require any additional computation. Then Table 4b compares several token compression options, including only pruning, only merging, pruning then merging, and merging then pruning. The results show that pruning then merging performs the best since it successfully combines the advantages of both pruning and merging. What's more, in Table 4c, we investigate the amount of training data required by DiffRate to find the optimal compression rate. Surprisingly, we find that only $1,000$ images are sufficient to obtain an appropriate compression rate. Although we also optimize the token compression rate using the entire training dataset, our results demonstrate the potential for DiffRate to work well even with minimal data. Lastly, Table 4d investigates the convergence time required by DiffRate. We can find that only three epochs are required, demonstrating the efficiency of DiffRate as a token compression approach.

6.4. Visualization

The visualizations results in Fig. 5 demonstrate that DiffRate effectively removes semantically irrelevant background information. Furthermore, DiffRate can reduce the number of tokens by merging less-discriminative tokens in the foreground. For example, in the first row, DiffRate successfully removes most of the background and merges the dog hair and butterfly wings tokens into fewer tokens. In the second row, DiffRate preserves salient information tokens in different image regions, even when multiple instances exist. Overall, the visualization results highlight the effectiveness of our proposed DiffRate method in compressing ViT models without significant information loss. See more results in Appendix D.

7. Conclusion

This work presents a new token compression framework, named Differentiable Compression Rate (DiffRate). The proposed approach integrates both token pruning and merging into a unified framework that can optimize the compression rate in a differentiable manner. To achieve this, we introduced a novel Differentiable Discrete Proxy (DDP) module that can effectively determine the optimal compression rate using gradient back-propagation. Our experimental results demonstrate that DiffRate can perform comparable or superior to previous state-of-the-art token compression methods, even without fine-tuning the model. Additionally, DiffRate is highly data-efficient, as it can identify the appropriate compression rate using only $1,000$ images.

Table 4: **Ablation experiments** using ViT-B (DeiT) [32]. Our default settings are marked in gray.

Metric	Acc.(%)
Random	79.72
$\mathbf{A_i}$	81.38
$\mathbf{A_c} \cdot \|\mathbf{V}\|$	**81.53**
$\mathbf{A_c}$	81.50

(a) **Sorting Metric.** Simply class attention can measure the importance of tokens.

Option	Acc.(%)
Pruning	80.83
Merging	81.14
Merging-Pruning	81.18
Pruning-Merging	**81.50**

(b) **Token Compression Module Option.** A merging and pruning pipeline is the best choice.

Number	Acc.(%)
1,000	81.40
4,000	81.46
16,000	**81.50**
All	**81.50**

(c) **Training Data.** 1,000 images is enough to optimize compression rate.

Time (g-hrs)	Acc.(%)
0.9 (1-ep)	81.32
2.7 (3-ep)	81.50
9 (10-ep)	81.50
27 (30-ep)	**81.52**

(d) **Optimization Time.** Token compression rate can converge within 2.7 gpu houres.

Overall, the proposed DiffRate framework offers a new perspective on token compression by revealing the importance of compression rate. We believe that this approach has the potential to pave the way for further advancements in token compression research.

Acknowledgement

This work is supported by National Key R&D Program of China (No.2022ZD0118201) , the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. U22B2051, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, No. 62002305 and No. 62272401), and the Natural Science Foundation of Fujian Province of China (No.2021J01002, No.2022J06001). This work is also partially supported by the National Key R&D Program of China(NO.2022ZD0160100), and in part by Shanghai Committee of Science and Technology (Grant No. 21DZ1100100).

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023. 1, 2, 3, 6, 7, 14

[2] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. *arXiv*, 2023. 17

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[4] Arnav Chavan, Zhiqiang Shen, Zhuang Liu, Zechun Liu, Kwang-Ting Cheng, and Eric P Xing. Vision transformer slimming: Multi-dimension searching in continuous optimization space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4931–4941, 2022. 2

[5] Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. Cf-vit: A general coarse-to-fine method for vision transformer. *arXiv preprint arXiv:2203.03821*, 2022. 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 14

[8] Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. *European Conference on Computer Vision (ECCV)*, 2022. 2, 7, 8

[9] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv preprint arXiv:2205.09113*, 2022. 1

[10] Hasan Genc, Seah Kim, Alon Amid, Ameer Haj-Ali, Vighnesh Iyer, Pranav Prakash, Jerry Zhao, Daniel Grubb, Harrison Liew, Howard Mao, Albert Ou, Colin Schmidt, Samuel Steffl, John Wright, Ion Stoica, Jonathan Ragan-Kelley, Krste Asanovic, Borivoje Nikolic, and Yakun Sophia Shao. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 769–774, 2021. 6, 13

[11] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1539–1547, 2020. 2

[12] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE TPAMI*, 2022. 1

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 1, 2, 6, 7, 13

[14] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012. 5

[15] Jie Hu, Liujuan Cao, Yao Lu, ShengChuan Zhang, Yan Wang, Ke Li, Feiyue Huang, Ling Shao, and Rongrong Ji. Istr: End-to-end instance segmentation with transformers. *arXiv preprint arXiv:2105.00637*, 2021. 1

[16] Jie Hu, Linyan Huang, Tianhe Ren, Shengchuan Zhang, Rongrong Ji, and Liujuan Cao. You only segment once: Towards real-time panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17819–17829, 2023. 1

[17] Ding Jia, Kai Han, Yunhe Wang, Yehui Tang, Jianyuan Guo, Chao Zhang, and Dacheng Tao. Efficient vision transformers via fine-grained manifold distillation. *arXiv preprint arXiv:2107.01378*, 2021. 1

[18] Zi-Hang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *Advances in neural information processing systems*, 34:18590–18602, 2021. 6

[19] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pages 5122–5131. PMLR, 2020. 3

[20] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pages 620–640. Springer, 2022. 2, 3, 7

[21] Ling Li, David Thorsley, and Joseph Hassoun. Sait: Sparse vision transformers through adaptive token pruning. *arXiv preprint arXiv:2210.05832*, 2022. 2

[22] Yanyu Li, Geng Yuan, Yang Wen, Eric Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *arXiv preprint arXiv:2206.01191*, 2022. 14

[23] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*, 2022. 1, 2, 3, 4, 6, 7, 8

[24] Mingbao Lin, Mengzhao Chen, Yuxin Zhang, Ke Li, Yunhang Shen, Chunhua Shen, and Rongrong Ji. Super vision transformer. *arXiv preprint arXiv:2205.11397*, 2022. 2

[25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021. 1, 14

[27] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 1

[28] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers. *arXiv preprint arXiv:2110.03860*, 2021. 2

[29] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 3, 4, 5, 7

[30] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021. 2

[31] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12165–12174, 2022. 2

[32] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021. 1, 6, 7, 9, 12, 13

[33] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 17

[34] Zhenyu Wang, Hao Luo, WANG Pichao, Feng Ding, Fan Wang, and Hao Li. Vtc-lfc: Vision transformer compression with low-frequency components. In *Advances in Neural Information Processing Systems*, 2022. 1, 2

[35] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI*, pages 68–85. Springer, 2022. 1

[36] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, pages 12077–12090, 2021. 1

[37] Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, Christoph Feichtenhofer, et al. Masked autoencoders that listen. *arXiv preprint arXiv:2207.06405*, 2022. 1

[38] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2964–2972, 2022. 2, 7

[39] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. *arXiv preprint arXiv:2110.04869*, 2021. 1

[40] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for

efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10809–10818, 2022. 2

[41] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision. *arXiv preprint arXiv:2210.13452*, 2022. 14, 17

[42] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers. *arXiv preprint arXiv:2111.12293*, 2021. 1

[43] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11101–11111, 2022. 2, 6, 7, 17

[44] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017. 17

[45] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2022. 1