Joint Token Pruning and Squeezing Towards More Aggressive Compression of Vision Transformers

Siyuan Wei[1*] Tianzhu Ye[2*] Shen Zhang[1] Yao Tang[1] Jiajun Liang[1†]
[1]MEGVII Technology [2]Tsinghua University
{weisiyuan, zhangshen,tangyao02,liangjiajun}@megvii.com,ytz20@mails.tsinghua.edu.cn

Abstract

Although vision transformers (ViTs) have shown promising results in various computer vision tasks recently, their high computational cost limits their practical applications. Previous approaches that prune redundant tokens have demonstrated a good trade-off between performance and computation costs. Nevertheless, errors caused by pruning strategies can lead to significant information loss. Our quantitative experiments reveal that the impact of pruned tokens on performance should be noticeable. To address this issue, we propose a novel joint Token Pruning & Squeezing module (TPS) for compressing vision transformers with higher efficiency. Firstly, TPS adopts pruning to get the reserved and pruned subsets. Secondly, TPS squeezes the information of pruned tokens into partial reserved tokens via the unidirectional nearest-neighbor matching and similarity-based fusing steps. Compared to state-of-the-art methods, our approach outperforms them under all token pruning intensities. Especially while shrinking DeiT-tiny&small computational budgets to 35%, it improves the accuracy by 1%-6% compared with baselines on ImageNet classification. The proposed method can accelerate the throughput of DeiT-small beyond DeiT-tiny, while its accuracy surpasses DeiT-tiny by 4.78%. Experiments on various transformers demonstrate the effectiveness of our method, while analysis experiments prove our higher robustness to the errors of the token pruning policy. Code is available at https://github.com/megvii-research/TPS-CVPR2023.

1. Introduction

The transformer architecture has become popular for various natural language processing (NLP) tasks, and its improved variants have been adopted for many vision tasks. Vision transformers (ViTs) [5] leverage the long-range dependencies of self-attention mechanisms to achieve excellent performance, often surpassing that of CNNs. In addition to the vanilla ViT architecture, recent studies [17, 31, 33] have explored hybrid ViT designs incorporating convolution layers and multi-scale architectures. Despite their excellent performance, transformers still require relatively high computational budgets. This is due to the quadratic computation and memory costs associated with token length. To address this issue, contemporary approaches [8, 14, 16, 21, 25, 27, 35, 36] propose pruning redundant tokens. They trade acceptable performance degradation for a more cost-effective model. Knowledge distillation [11] and other techniques can further mitigate the resulting performance drop.

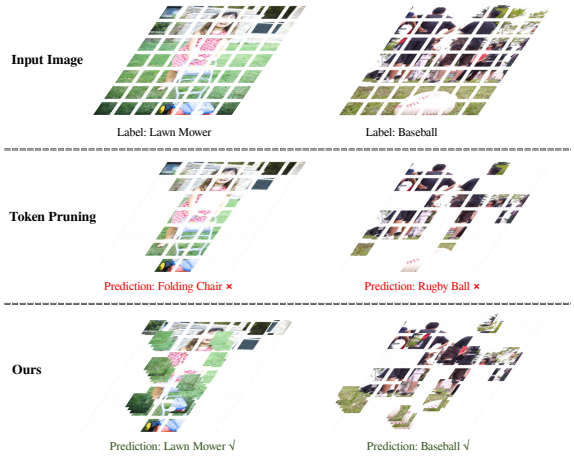


Figure 1. Comparisons between token pruning paradigm [25] (the 2nd row) and our joint Token Pruning & Squeezing (the 3rd row). The context information, such as the sod in the examples, is helpful for prediction but is discarded. Our method remits the information loss by squeezing the pruned tokens into reserved ones instead of naively dropping them, as indicated by the stacked patches. By this design, we could apply more aggressive token pruning with less performance drop. The example results are from the ImageNet1K [4], and we reduce the actual patches grid 14×14 to 7×7 for visualization clarity.

*The first two authors contributed equally to this work
†Corresponding author

However, a steep drop in performance is inevitable as pruning tokens further increases because both essential subject and auxiliary context information drop significantly, especially when the number of reserved tokens is closely below 10. Aggressive token pruning could lead to incomplete subject and background context loss, causing the wrong prediction, as shown in Fig. 1. Specifically, the background tokens containing sod help recognize the input image as a lawn mower rather than a folding chair. Meanwhile, missing subject tokens make the baseball indistinguishable from a rugby ball. To regain adequate information from pruned tokens, EViT [16] and Evo-ViT [35] propose aggregating pruned tokens as one, as shown in Fig. 2 (b). Still, they neglect the discrepancy among these tokens, leading to feature collapse and hindering more aggressive token pruning.

Towards more aggressive pruning, we argue that information in pruned tokens deserves better treatment. We did a toy experiment to answer what accuracy token pruning could achieve if it applied the reversed pruning policy in the first pruned transformer block as Fig. 3 shows. Taking dynamicViT [25] as a case study, the performance of reversed policy is enough to bring extra accuracy complementary to the original one (denoted by bonus accuracy). Moreover, this phenomenon would become more significant as pruning continues (red line in Fig. 3.).

To conserve the information from the pruned tokens, we propose a Joint Token Pruning & Squeezing (TPS) module to accommodate more aggressive compression of ViTs. TPS module utilizes a feature dispatch mechanism that squeezes essential features from pruned tokens into reserved ones, as shown in Fig. 2 (c). Firstly, based on the scoring result, the TPS module divides input tokens into two complementary subsets: the reserved and pruned sets. Secondly, instead of discarding or collapsing tokens from the pruned set into a single one, we employ a unidirectional nearest-neighbor matching algorithm to dispatch each of them independently to the associated reserved token dubbed as the host token. This design reduces information loss without sacrificing computational efficiency. Subsequently, we apply a similarity-based fusing way to squeeze the features of matched pruned tokens into corresponding host tokens while the non-selected reserved tokens remain identical. This design reduces the context information loss while retaining a reasonable computation budget. We can easily achieve hardware-friendly constant shape inference when fixing the cardinality of the reserved token set. Furthermore, we introduce two flexible variants: the inter-block version dTPS and the intra-block version eTPS, which are essentially plug-and-play blocks for both vanilla ViTs and hybrid ViTs.

We conduct extensive experiments on two datasets: ImageNet1K [4] and large fine-grained dataset iNaturalist 2019 [29] to prove our efficiency, flexibility, and robustness. Firstly, experiments under different token pruning settings demonstrate the superior performance of our TPS while operating more aggressive compression compared with token pruning [25] and token reorganization [16]; further comparisons with state-of-the-art transformers [8, 13, 20, 28, 31, 36, 39, 40] show our promising efficiency. Secondly, we manifest the flexibility of our TPS by integrating it into popular ViTs, including both vanilla ViTs and hybrid ViTs. Finally, the evaluations under the random token selection policy confirm the higher robustness of our TPS.

Overall, our contributions are summarized as follows:

- We propose the joint Token Pruning & Squeezing (TPS) and its two variants: dTPS and eTPS, to conserve the information of discarded tokens and facilitate more aggressive compression of vision transformers.
- Extensive experiments demonstrate our higher performance compared with prior approaches. Especially while compressing GFLOPs of DeiT-small&tiny to 35%, our TPS outperforms baselines with accuracy improvements of 1%-6%.
- Broadest experiments applying our method to vanilla ViTs and hybrid ViTs show our flexibility, while the analysis experiments prove that our TPS is more robust than token pruning and token reorganization.

2. Related Work

Since the transformer [30] was proved efficacious in NLP tasks, numerous studies have explored methods to acclimate the transformer architecture to computer vision tasks [1, 3, 5–7, 10, 17, 19, 22, 24, 26, 32, 37], including vanilla ViTs and hybrid ViTs.

Vanilla ViTs. Following the "primary ViT", a series of vision transformer variants inherit the central architecture and evolve from diverse perspectives, which we call the vanilla ViTs in this paper. DeiT [28] surpasses standard CNNs and ViT by introducing a distillation token to learn from a teacher network. LV-ViT [13] presents a new training objective called token labeling. T2T-ViT [38] recursively aggregates neighboring tokens into one token, while PS-ViT [39] introduces a progressive sampling module that selects informative tokens.

Hybrid ViTs. Besides, recent studies [15, 31, 33] incorporate convolutional layers and employ multi-scale architectures to lower the cost of computations and memory, which we call the hybrid ViTs. Swin Transformer [17] modified ViT with the multi-stage architecture and shifted window-based self-attention. CVT [33] presents a hierarchical architecture facilitated by the convolutional token embedding layer. PVT [31] introduces the pyramid architecture of the transformer and develops the spatial-reduction attention (SRA) to reduce the cost further.

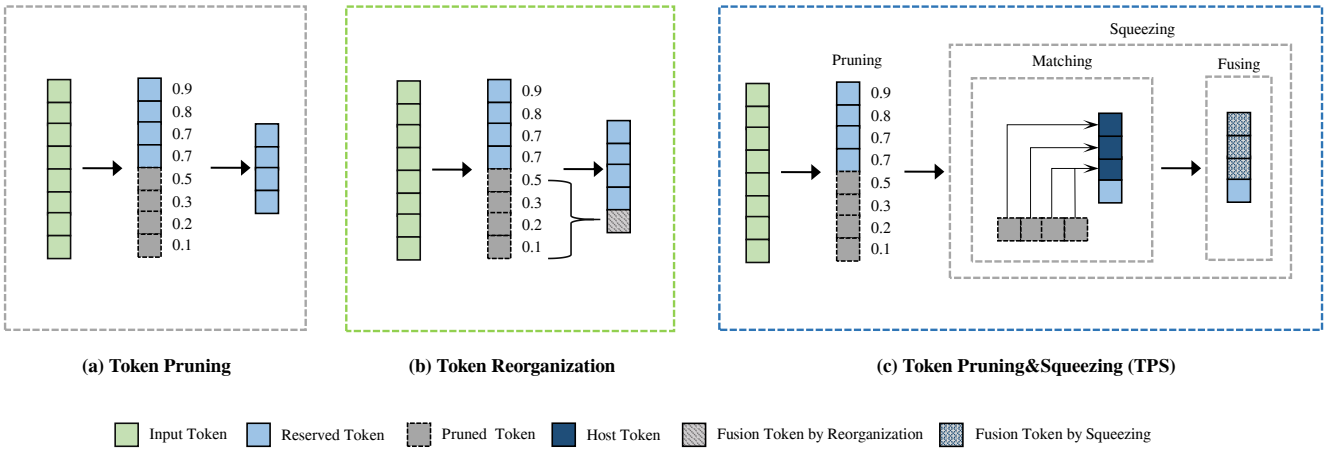


Figure 2. Comparison among token pruning [25], token reorganization [16], and our Token Pruning & Squeezing (TPS). Token pruning discards the pruned tokens; token reorganization aggregates pruned tokens into one without considering the discrepancy among them. To shrink tokens more efficiently, our TPS excavates the host token hiding in the reserved subset and squeezes similar pruned tokens into corresponding host tokens.

Token Pruning. Considering the spatial redundancy of input images, many researchers aim at discarding nonessential tokens with an acceptable performance drop. Tang *et al*. [27] propose to approximate the impact of patches and discard inattentive patches in a top-down paradigm. DynamicViT [25] and AdaViT [20] employ the learnable heads to score tokens and discard less informative ones with a fixed pruning ratio. A-ViT [36] and ATS [8] go further by sampling tokens with an input-dependent number. However, mainstream deep learning frameworks do not strongly support dynamic token length inference. The main disadvantage of token pruning models is the pruned information loss which leads to a drop in accuracy and limits more aggressive token pruning. To tackle this, Evo-ViT [35], EViT [16], and SPViT [14] preserve the background context by collapsing the pruned tokens into one token reorganization, which is called token reorganization. Token reorganization remits the pruned token information loss, but a noticeable performance drop can still be observed, especially regarding a higher pruning ratio of tokens. Furthermore, relevant auxiliary strategies are proposed to facilitate token pruning. SPViT [14] employs a layer-to-phase progressive training strategy, while IA-RED^2 performs a hierarchical training scheme. The complicated training schemes help improve performances but also draw into more hyperparameters and optimization difficulties.

We investigate the drawbacks of current token pruning methods and invent a novel token reduction approach: joint token Pruning & Squeezing with higher efficiency, robustness, and flexibility, which only requires fine-tuning pretrained models easily.

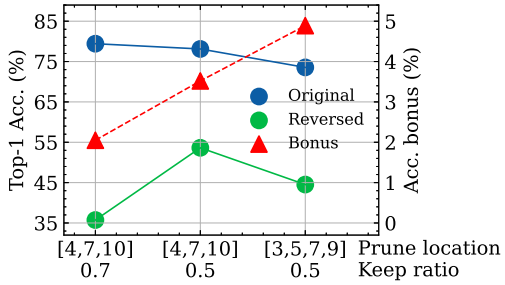


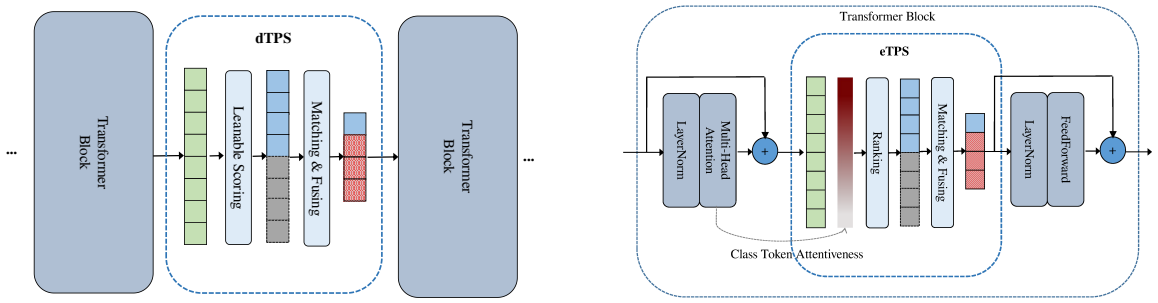
Figure 3. ImageNet1K results of DynamicViT [25] on DeiT-small [28] under the original policy and reversed policy. The reversed policy means exchanging reserved and pruned tokens in the first pruned layer. The right vertical axis implies the bonus accuracy dedicated by the cases that only the reversed policy predicts rightly.

3. Method

3.1. Motivation

To quantitatively verify the discarded information of pruned tokens, we conduct a toy experiment on DynamicViT [25] as Fig. 3 shows. It is easy to agree that the performance of pruned model declines as the pruning becomes more aggressive. Nevertheless, by exchanging reserved and pruned tokens (dubbed as the reversed policy in Fig. 3), we find that the pruned tokens can still handle partial cases correctly. Furthermore, the bonus accuracy, which is dedicated by the cases that only the reversed policy predicts rightly, rises along with the token pruning intensity. It implies that the exclusive information from pruned tokens matters more while the token pruning intensity grows.

These fun facts motivate us to assimilate the pruned to-



(a) The inter-block variant of our TPS: dTPS.

(b) The intra-block variant of our TPS: eTPS.

Figure 4. Two variants of our TPS. TPS can be plugged in both the inter-block and intra-block ways. For a fair comparison with DynamicViT [25] and EViT [16], our dTPS and eTPS adopt the same token scoring methods as the two baselines. The legend of tokens is the same as the Fig. 2.

kens into the reserved tokens to prevent information loss, as shown in Fig. 1. As shown in Fig. 2 (c), TPS employs two steps to compress ViTs, including token pruning and squeezing.

3.2. Token Pruning

In this section, we briefly review the basic procedure of token pruning. Note that our TPS is compatible with any token pruning techniques. Here, we introduce two variants of TPS: dTPS and eTPS, to cover both intra-block and inter-block token compression shown in Fig. 4. They follow the pruning parts of two baselines for a fair comparison with two typical baselines [16, 25].

As shown in Fig. 4, dTPS adopts the learnable token score prediction head from dynamicViT [25] and samples the binary decision mask by Straight-Through Gumbel-Softmax [12] for differentiability; eTPS utilizes the class token attention values to measure tokens' importance as EViT [16]. In the inference stage of both variants, based on token scores, we devise the token selection policy using the Top-k operation with a fixed given token reduction ratio ρ. Both variants ensure the constant shape to benefit from the inference optimization on the computation graph. The tokens are separated into two subsets, S^r and S^p, where the reserved tokens are placed in S^r and the pruned ones are placed in S^p. More implementation details can be found in our codes.

3.3. Token Squeezing

After reserved & pruned tokens are split, we introduce our token squeezing part. Considering that the reserved ones contribute the majority of correct predictions, we aim to design a procedure that retains most of the attentive tokens while compressing information from rest, preserving the model's overall performance. To avoid generating extra tokens as [14, 16], we inject pruned tokens into similar reserved tokens. So, we apply a unidirectional nearest-neighbor matching algorithm from S^p to S^r in a many-to-one manner. After that, we employ a similarity-based fusing method to assimilate information from pruned tokens into partial reserved tokens. We summarize the above process as two steps: *matching* and *fusing*.

Matching. Given the two subsets S^r and S^p, I^r and I^p are the corresponding token indices of S^r and S^p. A similarity matrix $c_{i,j}$ for all $i \in I^p$ and $j \in I^r$ represents the interactions between the tokens for matching. For each pruned token $\boldsymbol{x}_i \in S^p$, we find its nearest token $\boldsymbol{x}_*^{host} \in S^r$ from the reserved token set S^r as its *host token*:

$$\boldsymbol{x}_*^{host} = \underset{\boldsymbol{x}_j \in S^r}{argmax}\, c_{i,j}. \tag{1}$$

Note that since the token matching step is unidirectional from S^p to S^r, multiple pruned tokens can share the same host token and not each reserved token can serve as a host token. We then record the matching results in a mask matrix $\boldsymbol{M} \in \mathbb{R}^{N^p \times N^r}$ and its values are decided by:

$$m_{i,j} = \begin{cases} 1, & \boldsymbol{x}_j \; is \; the \; host \; token \; of \; \boldsymbol{x}_i, \\ 0, & otherwise, \end{cases} \tag{2}$$

where N^p and N^r denote the token number of two subsets. The mask helps that the following fusing step can be conducted with regular matrix operations on S^r and S^p while excluding the influence from non-matched pairs.

Although the attention map is a natural and free choice to measure interactions among tokens, we can acquire higher performances with the cosine similarity between S^r and S^p as the ablation experiment in Section. 4.2 discusses. Therefore in all of our experiments, the similarity matrix is defined as:

$$c_{i,j} = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|}, for \; i \in I^p, j \in I^r. \tag{3}$$

Since the similarity matrix $c_{i,j}$ is generated directly from input features, no extra parameters are introduced in the matching step.

Fusing. Simply averaging tokens can lead to feature dispersion because of discrepancies among the different tokens. EViT [16] utilizes the token importance scores to re-weight the aggregated tokens. Separately, we use a similarity-based weighting scheme. It expands the influence of closer tokens to the host tokens while also avoiding potential flaws from imperfect token scoring. As previously mentioned, the fusing step encompasses all tokens from two subsets and is controlled by the mask $\boldsymbol{M}$ to ensure that only host tokens and pruned tokens are mixed. This introduces a few redundant computations but increases practical training & inference throughput due to the efficiency of regular matrix operations.

Specifically, the reserved token $\boldsymbol{x}_j$ is updated by fusing the original feature and pruned tokens' features as follows:

$$\boldsymbol{y}_j = w_j \boldsymbol{x}_j + \sum_{\boldsymbol{x}_i \in S^p} w_i \boldsymbol{x}_i, \tag{4}$$

where w_i is the weight of each pruned token $\boldsymbol{x}_i \in S^p$, w_j is the weight of the reserved token itself, and $\boldsymbol{y}_j$ is the updated one. The fusing weight w_i depends on the mask value $m_{i,j}$ and similarity $c_{i,j}$:

$$w_i = \frac{\exp(c_{i,j}) m_{i,j}}{\sum_{\boldsymbol{x}_i \in S^p} \exp(c_{i,j}) m_{i,j} + \mathrm{e}}. \tag{5}$$

The reserved token always has the largest fusing weight w_j, as the similarity between x_j and itself equals to 1:

$$w_j = \frac{\mathrm{e}}{\sum_{\boldsymbol{x}_i \in S^p} \exp(c_{i,j}) m_{i,j} + \mathrm{e}}. \tag{6}$$

According to the above equations, the reserved tokens that have not been chosen as host tokens remain unchanged, while the pruned tokens are squeezed into host tokens and replace the original ones.

As can be seen, our matching and fusing steps ensure that the number of processed tokens equals the number of reserved tokens, thereby maintaining a constant shape for efficient inference.

3.4. TPS on Hybrid ViTs

To prove our flexibility and generalization across different transformers, we also conduct experiments in hybrid ViTs [31, 33]. For plain transformer blocks, our TPS modules can be easily inserted to reduce the token number and achieve a significant speedup. If the layer contains operations that require a complete spatial structured input: e.g., convolution or pooling, the operation of our TPS will be adjusted slightly. For example, in PVT [31] models, the TPS module is inserted before the first block of each stage with token pruning applied and generates the decision policy $\boldsymbol{D}$. For the attention layer, we decrease the token dimension size of the input and consequent query $\boldsymbol{Q}$. If the spatial-reduction layer is employed inside, the dropped token features are complemented with zeros to maintain the structured spatial input. More details can be found in supplementary materials.

4. Experiment

Datasets and evaluation metrics. We conduct contrast experiments with two typical baselines: DynamicViT [25] and EViT [16], and compare our performances with state-of-the-art transformers. For quantitative comparisons, we report the Top-1 accuracy, the number of giga floating-point operations (GFLOPs), and throughput. The input size is set to 224×224 for all the experiments. The evaluated datasets include the ImageNet1K [4] and the large fine-grained image classification dataset: iNaturalist 2019 [29].

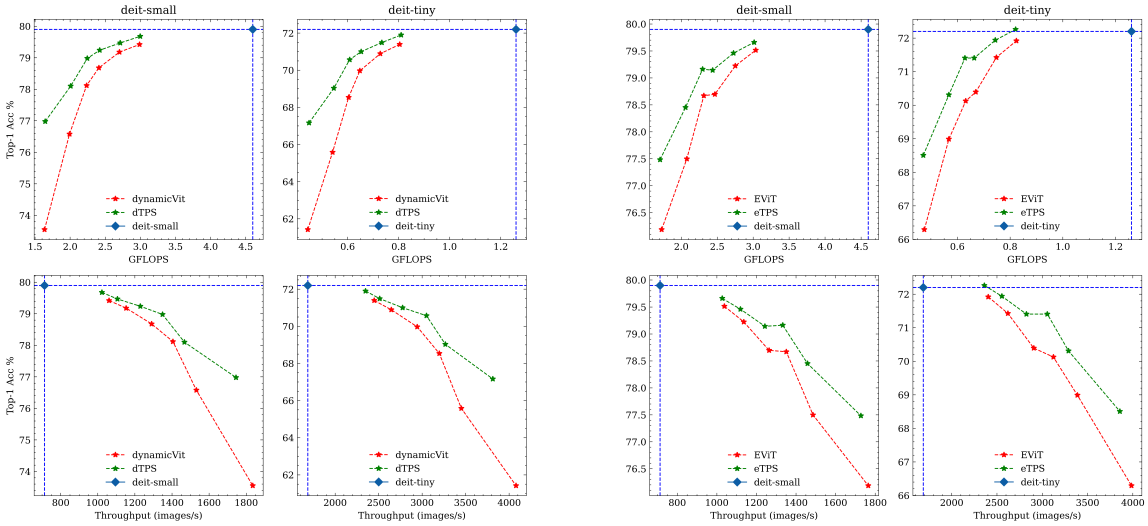
Experiments Details. We follow the same data augmentations used in DeiT [28]. The model is initialized with pre-trained models' weights and fine-tuned with different token pruning locations and keeping ratios. We adopt the AdamW [18] as the optimizer and a cosine learning rate scheduler. We compare our dTPS and eTPS with DynamicViT [25] and EViT [16] under multiple pruning settings[1]. We follow the same training settings and loss functions from [16, 25], except for the basic learning rate is set to $\frac{batchsize}{1024} \times 2.5 \times 10^{-4}$ and no stage of fixing backbone weights in dynamicViT & dTPS. The setting changes slightly because the training under the original setting appears unstable, especially with aggressive pruning.

4.1. Main Results

Comparison to baselines. As Fig. 5 shows, we compare our method with the token pruning baseline: DynamicViT, and the token reorganization baseline: EViT, by replacing their original pruning modules with our dTPS & eTPS modules. The contrast experiments involve DeiT-small&tiny. All the models in this part are fine-tuned 30 epochs under multiple pruning settings. Under all the settings, our method outperforms DynamicViT and EViT. Both dynamicViT and EViT encounter a larger accuracy drop along with the progressively aggressive pruning. While shrinking the computational budgets of DeiT to 35%, our method can avoid **1%-6%** accuracy decline compared with baselines. Equipped with our TPS, we can accelerate the throughput of DeiT-small to **1745 images/s**, which is beyond that of DeiT-tiny: **1686 images/s**, and surpass the accuracy of DeiT-tiny by **4.78%**.

Visual Comparisons. We demonstrate the cases from ImageNet1K [4], which DeiT predicts correctly at first but gives the wrong predictions after being applied with token pruning. As Fig. 7 shows, the imperfect pruning policy

[1]The pruning settings include combinations of three multi-layer pruning settings: pruning locations include $\{4^{\text{th}},7^{\text{th}},10^{\text{th}}\}$,$\{3^{\text{rd}},5^{\text{th}},7^{\text{th}},9^{\text{th}}\}$, and $\{4^{\text{th}},6^{\text{th}},8^{\text{th}},10^{\text{th}}\}$, and token keeping ratios $\rho \in \{0.5, 0.7\}$



(a) Comparison between our dTPS and dynamicViT on DeiT.

(b) Comparison between our eTPS and EViT on DeiT.

Figure 5. ImageNet1K results of our two variants: dTPS & eTPS, and two baselines: DynamicViT [25] & EViT [16], under different GFLOPs of pruned DeiT-small&tiny [28]. The parameter number of two variants is the same as the two baselines respectively. The throughput is measured on a single NVIDIA RTX 2080Ti with a batch size of 32. The more aggressively we apply token pruning on backbones, the more competitive accuracy-computation trade-off our method shows. See supplementary materials for TPS on DeiT-base and with a 384×384 input size .

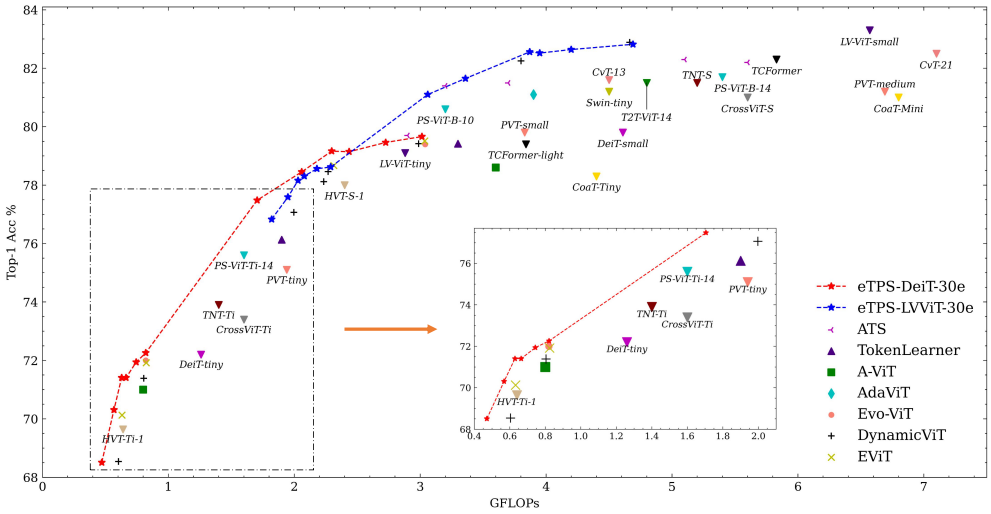


Figure 6. Comparison between DeiT & LV-ViT with our TPS applied and other transformer methods, including token pruning [8, 9, 16, 20, 20, 25, 35], vanilla ViTs [2, 13, 23, 28, 38, 39] and hybrid ViTs [17, 31, 34, 40]. Our TPS outperforms numerous state-of-the-art transformers on ImageNet1K image classification with only 30 fine-tuning epochs required.

brings the context information loss, which leads to a close but incorrect prediction. However, our TPS remedies these cases by saving the pruned tokens' information.

Comparison to states of the art. In Fig. 6, we demonstrate our TPS performances compared with other state-of-the-art transformers, including token pruning methods [8, 16, 20, 20, 25, 35], vanilla ViTs [2, 13, 23, 28, 38, 39] and hybrid ViTs [17, 31, 34, 40]. By integrating DeiT-small&tiny and LV-ViT-small&tiny with TPS and fine-tuning them only 30 epochs, we can achieve a quite competitive performance among numerous vision transformers from the perspective of accuracy-computation trade-off.

Extension on more backbones. As shown in Tab. 1 and Tab. 2, we incorporate TPS into different vanilla ViTs [28, 38, 39] and hybrid ViTs [31, 40] to prove the flexibility and generalization. For vanilla ViTs, our TPS out-

Figure 7. Visual comparisons between token pruning [25] and our TPS. DynamicViT and our dTPS on DeiT-small give the results. The pruning locations include $\{4^{th}, 7^{th}, 8^{th}\}$ and each pruning stage's keeping ratio is 0.5. The cases are displayed based on the results of the first pruning stage. For DynamicViT, the blank areas denote pruned tokens; for dTPS, we mask each group of matched tokens as the same color for visualization clarity.

performs EViT [16], Evo-ViT [35], A-ViT [36], IA-RED^2 and SPViT [14] with equal or slightly increasing computation while using DeiT [28], LV-ViT [13] as backbones. DeiT-small&tiny with TPS applied can surpass the pre-trained models by **0.3%** and **0.7%** in accuracy under 100 fine-tuning epochs. For hybrid ViTs, we can compress the GFLOPS of PVT-tiny by 13% and improve its accuracy by 0.1%.

Fine-Grained Visual categorization. We compare our dTPS with DynamicViT by fine-tuning DeiT on iNaturalist 2019 [29] as shown in Tab. 3. See the supplementary materials for the training details on iNaturalist 2019 [29]. Compared with dynamicViT, our dTPS obtains 0.3% accuracy improvement in DeiT-tiny and 0.2% accuracy improvement in DeiT-small when fine-tuning 30 epochs, respectively. We further fine-tune dTPS 100 epochs and observe a significant improvement in both backbones. Notably, dTPS fine-tuned with 100 epochs is only 0.1% lower than Deit-small while shrinking the computational budgets of Deit-small to 65%.

4.2. Ablation Study

Epochs of training. Fig. 8 shows that both variants can benefit from longer training epochs and surpass the DeiT-small&tiny with only 65% GFLOPS. However, the benefit of epochs varies slightly in two variants. Because the class-token attention scoring requires no extra optimization target, eTPS performs better than dTPS under 30 epochs. On the other hand, dTPS can benefit more from longer training epochs in DeiT-small, for its learnable scoring brings higher performance upper bound.

Feature Type. We show the effects of feature type used to establish the matching relationships. Supposing $\boldsymbol{x}_i$ is the full embedding of the token and the position feature $\boldsymbol{p}_i$ is the corresponding positional embedding, we define the content feature as $\boldsymbol{x}_i - \boldsymbol{p}_i$. As Tab. 4 illustrates, the entire feature is more favorable for it contains both the content and position information.

Method	Param(M)	GFLOPs	Top-1 Acc.(%)
DeiT-S	22.05	4.6	79.8
DynamicViT [25]	22.77	2.9	79.3
EViT [16]	**22.05**	3.0	79.5
ATS† [8]	**22.05**	2.9	**79.7**
A-ViT† [36] (100 epochs)	**22.05**	3.6	78.6
Evo-ViT [35] (300 epochs)	**22.05**	3.0	79.4
SPViT [14] (75 epochs)	22.13	**2.7**	79.3
IA-RED^2 [21] (90 epochs)	-	-	79.1
eTPS (ours)	**22.05**	3.0	**79.7**
dTPS* (ours)	22.77	3.0	**80.1**
DeiT-T	**5.72**	1.3	72.2
DynamicViT(re-impl) [25]	5.90	**0.8**	71.4
EViT(re-impl) [16]	5.72	**0.8**	71.9
A-ViT† [36] (100 epochs)	**5.00**	**0.8**	71.0
Evo-ViT [35] (300 epochs)	5.72	**0.8**	72.0
SPViT [14] (75 epochs)	-	0.9	72.1
eTPS (ours)	5.72	**0.8**	**72.3**
dTPS* (ours)	5.90	**0.8**	**72.9**
LV-ViT-S	26.17	6.6	83.3
DynamicViT [25]	26.89	**3.8**	82.0
EViT [16]	**26.17**	3.9	**82.5**
eTPS (ours)	**26.17**	**3.8**	**82.5**
dTPS* (ours)	26.89	**3.8**	**82.6**
LV-ViT-T	8.53	2.9	79.1
DynamicViT(re-impl) [25]	8.82	2.0	77.1
eTPS (ours)	**8.53**	**2.0**	**78.0**
dTPS* (ours)	**8.82**	**2.0**	**78.7**
PS-ViT-B/14 [39]	21.34	5.4	81.7
ATS† [8]	**21.34**	**3.7**	**81.5**
dTPS* (ours)	22.07	**3.7**	**81.5**

Table 1. Comparison among different token pruning methods applied to multiple vanilla vision transformers. "*" denotes our method is fine-tuned 100 epochs. Methods marked with "†" do not support constant-shape inference. Prior methods above are trained 30 epochs by default unless otherwise specified. "Re-impl" means that we implement the method according to the official code. For a fair comparison with prior methods, we utilize computationally comparable pruning setups to fine-tune backbones with TPS.

Method	Param (M)	GFLOPs	Top-1 Acc. (%)
PVT-T [31]	13.23	1.94	75.1
dTPS* (ours)	13.85	**1.69 (-13%)**	**75.2 (+0.1)**
PVT-S	24.49	3.83	79.8
dTPS* (ours)	25.11	**3.14 (-18%)**	79.2 (-0.6)
CvT-13 [33]	20.00	4.58	81.6
dTPS* (ours)	20.72	**3.04 (-34%)**	80.8 (-0.8)
CvT-21	31.62	7.21	82.5
dTPS* (ours)	32.35	**4.10 (-43%)**	80.9 (-1.6)

Table 2. Experiments of our methods applied to hybrid vision transformers, including PVT [31] and CVT [33].

Method	Param(M)	GFLOPs	Top-1 Acc.(%)
DeiT-S [28]	22.05	4.6	74.8
DynamicViT(re-impl) [25]	**22.77**	**2.9**	74.0
dTPS (ours)	**22.77**	3.0	74.2
dTPS* (ours)	**22.77**	3.0	**74.7**
DeiT-T	5.72	1.26	72.8
DynamicViT(re-impl) [25]	**5.90**	**0.8**	71.4
dTPS (ours)	**5.90**	**0.8**	71.7
dTPS* (ours)	**5.90**	**0.8**	**72.4**

Table 3. Results of dynamicViT [25] and dTPS on iNaturalist 2019 [29]. The two models are trained 30 epochs by default. "*" denotes the model is trained 100 epochs. "Re-impl" means we implement the method on the backbone according to the official code. See the appendix for the training details of the backbone.

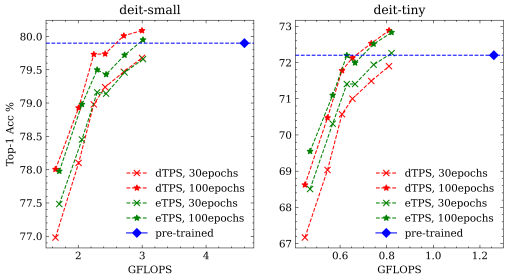


Figure 8. ImageNet1K results of our dTPS & eTPS under different GFLOPs of the pruned model under 30 & 100 fine-tuing epochs. Both dTPS and eTPS benefit from longer epochs and surpass the performances of pre-trained DeiT-small&tiny.

Feature Type	Top-1 Acc. (%)
Full	**71.90**
Content	71.73
Position	70.92

Table 4. Different feature types used in the matching step. Conducted on DeiT-tiny with pruned layers at $\{4^{th},7^{th},8^{th}\}$ and keeping ratio set to 0.7.

TPM Variant	Similarity Matrix	GFLOPs	Top-1 Acc.(%)
dTPS	Cosine similarity	0.810	**71.90**
	Previous attention	0.807	71.35
eTPS	Cosine similarity	0.821	**72.26**
	Previous attention	0.818	71.67

Table 5. Comparison between different types of cost matrix. The baseline denotes calculating the tokens' cosine similarity; the previous attention denotes that we reuse it to devise the matching results. Conducted on DeiT-tiny with pruned layers at $\{4^{th},7^{th},8^{th}\}$ and keeping ratio set to 0.7.

Similarity Matrix. Considering that the dot-product attention of query and key measures tokens' relationships naturally, we have tried reusing the previous attention to replace the computations of cosine similarities in the matching step. We believe the previous attention is outdated to measure current tokens' relations and Tab. 5 shows that calculating the cosine similarity of current features outperforms reusing the attention with only a minor computational increase.

Methods	Policy	Top-1 Acc. (%)
DynamicViT	Original	79.42
	Random	76.51 (-3.7)
dTPS	Original	79.68
	Random	78.19 (**-1.9**)
EViT	Original	79.51
	Random	77.47 (-2.6)
eTPS	Original	79.66
	Random	78.06 (**-2.0**)

Table 6. ImageNet1K results of applying random token selection policy to our methods and baselines. The percentages in parentheses represent the relative performance degradation ratio brought by random policies.

4.3. Robustness Experiments

We generate random token selection policies to construct manufactured policy errors that simulate the cases brought by sub-optimal token pruning strategies. All models are based on DeiT-small and fine-tuned 30 epochs with identical pruning setups. We run the experiments under random policies 100 times and report the average results. By comparing the performances of our method with dynamicViT [25] and EViT [16], the accuracy drop from the original to random policies denotes the robustness under incorrect policies. As shown in Tab. 6, our inter-block version dTPS and intra-block version eTPS have fewer accuracy drops than dynamicViT [25] and EViT [16].

5. Conclusions and Limitations

In this paper, we presented a novel joint Token Pruning & Squeezing (TPS) module to compress vision transformers more aggressively. With the capability of conserving information, our TPS can avoid a significant performance drop compared to token pruning and reorganization. Our method has better efficiency than prior token pruning methods and states of the arts in vision transformers. Extensive experiments under various backbones and quantitative analyses show our flexibility and robustness.

However, there are still some limitations to our method. Firstly, structured spatial operations of hybrid ViTs restrict the straightforward integration of token pruning. Secondly, the procedure of fine-tuning pre-trained models might be replaced by more advanced pruning-aware training-from-scratch schemes to shorten the total training time. In the future, we will evolve our method to be more adaptive to hybrid ViTs and apply it to more dense prediction tasks.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[2] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021. 6

[3] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021. 2

[4] 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. 1, 2, 5

[5] 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. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2

[6] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021. 2

[7] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021. 2

[8] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei, and Sommerlade1 Hamed Pirsiavash2 Juergen Gall. Adaptive token sampling for efficient vision transformers. 1, 2, 3, 6, 7

[9] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021. 6

[10] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15013–15022, 2021. 2

[11] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 1

[12] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 4

[13] 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. 2, 6, 7

[14] 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 *European Conference on Computer Vision*, pages 620–640. Springer, 2022. 1, 3, 4, 7

[15] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021. 2

[16] 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. *arXiv preprint arXiv:2202.07800*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[17] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 2, 6

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[19] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021. 2

[20] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12309–12318, 2022. 2, 3, 6

[21] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red ˆ2: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34:24898–24911, 2021. 1, 7

[22] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7463–7472, 2021. 2

[23] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/cvf international conference on computer vision*, pages 377–386, 2021. 6

[24] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021. 2

[25] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[26] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *Advances in Neural Information Processing Systems*, 34:980–993, 2021. 2

[27] 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. 1, 3

[28] 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*, pages 10347–10357. PMLR, 2021. 2, 3, 5, 6, 7, 8

[29] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist challenge 2019 dataset. *arXiv preprint arXiv:1707.06642*, 2019. 2, 5, 7, 8

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[31] 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. 1, 2, 5, 6, 7

[32] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021. 2

[33] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021. 1, 2, 5, 7

[34] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9981–9990, 2021. 6

[35] 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. 1, 2, 3, 6, 7

[36] 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. 1, 2, 3, 7

[37] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12498–12507, 2021. 2

[38] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021. 2, 6

[39] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip HS Torr, Wayne Zhang, and Dahua Lin. Vision transformer with progressive sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 387–396, 2021. 2, 6, 7

[40] 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