SPViT: Enabling Faster Vision Transformers via Latency-aware Soft Token Pruning

Zhenglun Kong*[1], Peiyan Dong*[1], Xiaolong Ma[2], Xin Meng[3], Wei Niu[4], Mengshu Sun[1], Xuan Shen[1], Geng Yuan[1], Bin Ren[4], Hao Tang[5], Minghai Qin[1], and Yanzhi Wang[1]

[1] Northeastern University, Boston MA 02115, USA
{kong.zhe,dong.pe,yanz.wang}@northeastern.edu
[2] Clemson University, Clemson SC 29634, USA
[3] Peking university, Beijing 100871, China
[4] College of William and Mary, Williamsburg VA 23185, USA
[5] CVL, ETH Zürich, Zürich 8092, Switzerland

Abstract. Recently, Vision Transformer (ViT) has continuously established new milestones in the computer vision field, while the high computation and memory cost makes its propagation in industrial production difficult. Considering the computation complexity, the internal data pattern of ViTs, and the edge device deployment, we propose a latency-aware soft token pruning framework, **SPViT**, which can be set up on vanilla Transformers of both flatten and hierarchical structures, such as DeiTs and Swin-Transformers (Swin). More concretely, we design a dynamic attention-based multi-head token selector, which is a lightweight module for adaptive instance-wise token selection. We further introduce a soft pruning technique, which integrates the less informative tokens chosen by the selector module into a package token rather than discarding them completely. SPViT is bound to the trade-off between accuracy and latency requirements of specific edge devices through our proposed latency-aware training strategy. Experiment results show that SPViT significantly reduces the computation cost of ViTs with comparable performance on image classification. Moreover, SPViT can guarantee the identified model meets the latency specifications of mobile devices and FPGA, and even achieve the real-time execution of DeiT-T on mobile devices. For example, SPViT reduces the latency of DeiT-T to 26 ms (26%∼41% superior to existing works) on the mobile device with 0.25%∼4% higher top-1 accuracy on ImageNet. Our code is released at https://github.com/PeiyanFlying/SPViT

Keywords: Vision Transformer; Model Compression; Hardware Acceleration; Mobile Devices; FPGA

1 Introduction

Recently, a new trend of leveraging Transformer architecture [80] into the computer vision domain has emerged [38,13,41,21,90,109,33,75]. The Vision Trans-

* Both authors contributed equally.

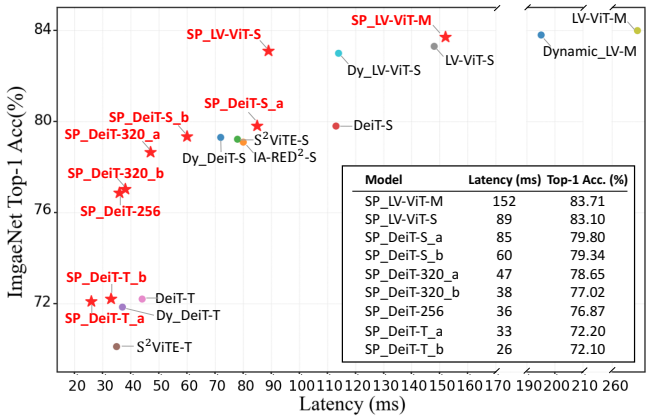


Fig. 1: Comparison of different pruning methods with various accuracy-latency trade-offs. We can increase the accuracy of light weight models at similar latency, and expedite larger models with negligible decrease of accuracy. Models are tested on Samsung Galaxy S20.

former (ViT), which solely exploits the self-attention mechanism that inherits from the Transformer architecture, has set up many state-of-the-art (SOTA) records in image classifications [22,79,7], object detection [3,19,1,60], tracking [15,91,59], semantic segmentation [110,16], depth estimation [94,45], image retrieval [23], and image enhancement [93,8,50]. However, despite the impressive general results, ViTs have sacrificed lightweight model capacity, portability, and trainability in return for high accuracy. The mass amount of computations brought by operations (e.g. Conv, MatMul, Add) in existing models remains a setback for edge device deployment.

Pruning has been proved as the one of the most effective methods to reduce network dimensions in convolution-based neural networks [70,101,57,47,107,62,72,52,108,11,36,55,54,4]. However, when huge amount of AI-powered applications are benefiting from the network pruning advantages [63,51,53,99,98,30,77,17,56,100,44,26,25,27], the applications of self-attention-based neural network pruning remain scarce [32,74,43,81,61]. There still exists a gap between the actual device deployment and acceleration in the ViT pruning frameworks. For instance, attention head pruning [12] performs weight pruning on the transformation matrix (W_Q, W_K, W_V) before the multi-head self-attention (MSA) operation. It is an inefficient way for computation reduction because only part of the ViT computations (i.e., MSA) can be alleviated (see Sec. 3 for justification). In a lightweight model, head pruning cannot guarantee an ideal pruning rate without significant accuracy deterioration. Static token pruning [69] reduces the number of input tokens by a fixed ratio for different images, which restricts the image pruning rate, ignoring the fact that the high-level information of each image varies both in the region size and location. Furthermore, it is difficult for the deployment on edge devices since

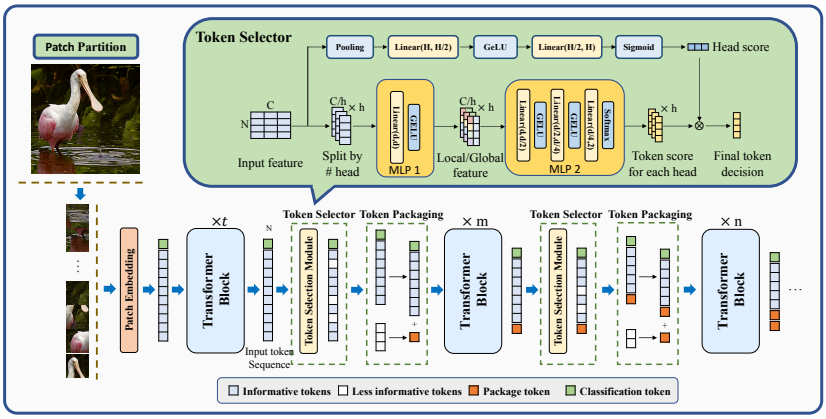


Fig. 2: Overall workflow. Bottom figure: Token selector is inserted multiple times throughout the model, along with the token packaging technique to generate a package token from the less informative tokens. The package token is concatenated with the informative tokens to be fed in the following transformer blocks. Upper figure: Our attention-based multi-head token selector to obtain token scores for keep/prune decisions.

newly introduced operations (e.g., Argsort) are currently not well supported by many frameworks [66]. In contrast, dynamic token pruning [64] deletes redundant tokens based on the inherent image characteristics to achieve a per-image adaptive pruning rate. However, this method implies a potentially huge search space, which will easily cause a limited overall pruning rate or undermined accuracy if the token selection mechanism is not carefully designed. In addition, the pruning mechanism in [64] unreservedly discards less informative tokens, which results in the loss of the informative part of the removed tokens.

In this paper, we manage to overcome the above limitations. Specifically, as shown in Fig. 2, we propose a latency-aware Soft Pruning framework (SPViT), which simultaneously optimizes ViT accuracy and maximizes per-image dynamic pruning rate while maintaining actual computation constraints on edge devices. In ViT, each head encodes the visual receptive field independently [64,35,58], which implies that each token has a different influence in different heads [22,106,96,29]. We thus propose a token selector to evaluate the importance score of each token based on its characteristic statistics in all heads. Then, through an attention-based branch [37] in the selector, we calculate the weighted sum of each score to obtain the final score of a token, which determines whether the token should be pruned. With the token selector, all tokens generated from the input images can be precisely ranked and pruned based on their importance scores and thus achieving a high overall pruning rate.

The token representations [84,87,10,5] in early and middle layers are insufficiently encoded, which makes token pruning quite difficult. To mitigate the challenge, we introduce a package token technique, which compresses the less-

informative tokens, picked out by the token selector, into a package token. Then, we concatenate the package token to the remaining tokens for subsequent blocks. On the one hand, although informative tokens may be discarded due to the poor encoding ability in earlier blocks of ViT [89], this error will be partly corrected by the residual information stored in the package token. On the other hand, background features can help emphasize foreground features [92]. Completely removing less informative (negative) tokens will weaken the ability of self-attention to capture key information. Therefore, the package token can serve as a way to help preserve background features. By adding minimal computation cost, the token pruning rate can be increased significantly.

In addition, we elaborate a latency-aware training strategy, which consists of two parts: latency-aware loss function and layer-to-phase progressive training. The former bridges the token pruning rates with latency specifications of diverse edge devices. The latter indicates that we progressively insert one selector in each block and train the new selector under the latency budget of the target device. Next, we group adjacent blocks with similar pruning rates into a phase, keep the first selector in this phase and remove others. While maintaining high accuracy, it can search for the appropriate pruning rate for each block and the desired insertion position of the selector. Fig. 1 shows the on device performance of our model compared with other pruned or scaled models.

Our contributions are summarized as follows:

- We provide a detailed analysis on the computational complexity of ViT and different compression strategies. Based on our analysis, token pruning holds a greater computation reduction compared to the compression of other dimensions.
- Considering the vision pattern inside ViT, we propose SPViT, a novel method which includes the attention-based multi-head token selector and the token packaging technique to achieve per-image adaptive pruning. We design a latency-aware training strategy, which efficiently explores the SPViT design space given the hardware latency budget, and maximizes the per-image pruning rate without any accuracy degradation.
- SPViT enables a higher pruning rate than other state-of-the-art with comparable accuracy. For lightweight models, SPViT allows the DeiT-S and DeiT-T to reduce inference latency by 40%-60% within 0.5% accuracy loss. It can further generate more efficient PiTs and Swins with negligible performance drops. In particular, SPViT is superior in the compression of lightweight models.
- We demonstrate a real-time realization of DeiT-T on mobile phones (e.g., 26 ms on a Samsung Galaxy S20) and DeiT-S on a Xilinx FPGA (13.2 ms on a Xilinx ZCU102). To the best of our knowledge, it is the first time that the ViT models perform inference on the edge devices beyond real-time[6].

[6] Real-time inference usually means 30 frames per second, which is approximately 33 ms/image.

2 Related Work

Vision Transformers. ViT [22] is a pioneering work that uses only a Transformer to solve various vision tasks. Compared to traditional CNN structures, ViT allows all the positions in an image to interact through transformer blocks, whereas CNNs operate on a fixed-sized window with restricted spatial interactions, which can have trouble capturing relations at the pixel level in both spatial and time domains [68]. Since then, many variants have been proposed [31,49,102,83,34,86,9,76,24,48,82,2]. For example, DeiT [79], T2T-ViT [103] and Mixer [14] tackle the data-inefficiency problem in ViT by training only with ImageNet. PiT [35] replaces the uniform structure of Transformer with depth-wise convolution pooling layer to reduce spacial dimension and increase channel dimension. LV-ViT [40] introduces a token labeling method to improve training. PS-ViT [105] applied progressive sampled tokens.

Efficient ViT. The huge memory usage and computation cost of the self-attention mechanism serve as the roadblock to the efficient deployment of ViT models on edge devices. Many works aim at accelerating the inference speed of ViT [6]. For instance, S^2ViTE [12] prunes token and attention head in a structured way via sparse training. VTP [112] reduces the input feature dimension by learning their associated importance scores with L1 regularization. IA-RED^2 [64] drops redundant tokens with a multi-head interpreter. PS-ViT (T2T) [78] discards useless patches in a top-down paradigm. DynamicViT [69] removes redundant tokens by estimating their importance score with a MLP [80] based prediction module. Evo-ViT [89] develops a slow-fast token evolution method to preserve more image information during pruning. TokenLearner [73] and PATCHMERGER [71] uses spatial attention to generate a small set of token vectors adaptive to the input. However, to the best of our knowledge, our idea of considering actual edge device deployment and acceleration has not been investigated by any existing ViT pruning methods.

3 Computational Complexity Analysis

Given an input sequence $N \times D$, where N is the input sequence length or the token number and D is the embedding dimension [79] of each token, some works [64,112] address the computational complexity of ViT as $(12ND^2+2N^2D)$. However, D represents different dimensions and should be written as $(4ND_{ch}D_{attn}+2N^2D_{attn}+8ND_{ch}D_{fc})$. Neglecting the difference may cause misleading conclusions, especially when analyzing the validity of pruning methods such as token pruning and dimension pruning.

Table 1 shows an analysis of each operation in a Transformer block. There are three main branches of ViT pruning. (i) Token channel pruning: The sequence tokens are pruned along D_{ch} dimension. D_{ch} is non-transmissible, which means reducing input dimension only affects the computation of the current matrix multiplication. To reduce computation for all layers, a mask layer is added to multiply with the input before going through the linear layer [112]. (ii) Token

Table 1: The computational complexity of each operation in a ViT block. The input $N \times D_{ch}$ goes through three linear transformation layers with $D_{ch} \times D_{attn}$ to generate Query (Q), Key (K), and Value (V) matrices of size $N \times D_{attn}$. N is transitive, while D_{ch} is not.

#	Module	Input Size	Operation	Layer Size	Output Size	Computation
①	MSA	$N \times D_{ch}$	Linear Transformation	$D_{ch} \times D_{attn}$	$N \times D_{attn}$	$ND_{ch}D_{attn} \times 3$
②		$N \times D_{attn}$	Q Multiplying K^T	-	$N \times N$	$N^2 D_{attn}$
③		$N \times N$	Multiplying V	-	$N \times D_{attn}$	$N^2 D_{attn}$
④		$N \times D_{attn}$	Projection	$D_{attn} \times D_{ch}$	$N \times D_{ch}$	$ND_{attn}D_{ch}$
⑤	FNN	$N \times D_{ch}$	FC Layer	$D_{ch} \times 4D_{fc}$	$N \times 4D_{fc}$	$4ND_{ch}D_{fc}$
⑥		$N \times 4D_{fc}$	FC Layer	$4D_{fc} \times D_{ch}$	$N \times D_{ch}$	$4ND_{fc}D_{ch}$
Total Computational Complexity						$4ND_{ch}D_{attn} + 2N^2 D_{attn} + 8ND_{ch}D_{fc}$

pruning: N is transitive, so directly pruning tokens will contribute to the linearly or even quadratically (N^2 in ② and ③) reduction of all operations. (iii) Attention head pruning (or attention channel pruning): The pruning operations are performed on weight tensors of each attention head in the MSA module. However, only the D_{attn} in the MSA module can be counted towards computation reduction, which usually contributes less than 40% of the total computation in most ViT architectures. Therefore, with the same pruning rate, pruning tokens (reducing N) can reduce more overall computation than pruning channels (reducing D_{ch} or D_{attn}).

4 Latency-Aware Soft Pruning

In this section, we first introduce our soft token pruning framework. Then, we show an elaborate design of each module. Finally, we give a detailed discussion of our latency-aware training strategy.

4.1 Framework Overview

Our soft pruning framework includes a token selector and a token packaging technique. We propose a hierarchical pruning scheme, where these two modules are inserted between multiple blocks throughout the model. As shown in Fig. 2, the input token sequence first goes through a token selector, where each token is scored and defined as either informative or less informative. After that, less informative tokens are separated from the sequence and integrated into a package token. This package token then concatenates to the informative tokens to involve in subsequent calculations in the blocks. In the next phase, a newly generated package token will connect with the existing package token.

For ViT training with our framework, we devise a latency-aware sparsity loss for the hardware's maximum computation bandwidth. We perform a layer-to-phase progressive training schedule to compress the search space, where model

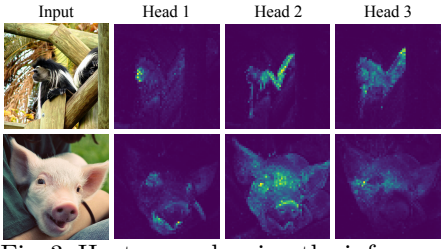


Fig. 3: Heatmaps showing the informative region detected by each head in DeiT-T. Each attention head focuses on encoding different image features and visual receptive fields.

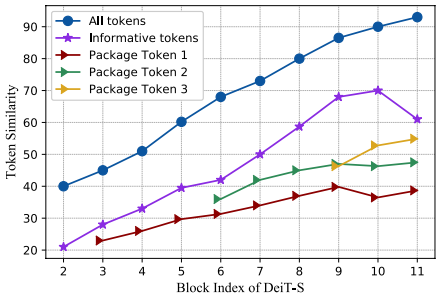


Fig. 4: The CKA between the final CLS token and other tokens.

accuracy optimization and hardware computation reduction can be simultaneously achieved. The overall framework is hardware friendly with no unsupported operations and miniature computation cost.

Multi-head Token Selector. We propose a fine-grained approach to evaluate token scores. As shown in Fig. 3, in ViT's multi-head vision pattern, each head focus on encoding different features and respective fields of an image. This implies that the importance of each token towards each head is different. Our multi-head selector generates a list of token scores for each head. Let one head dimension be $d{=}C/H$, where C is the input dimension and H is the number of head. We split the input $X{\in}\mathbb{R}^{N\times C}$ by the number of attention head into $\{x_i\}_{i=1}^{H}{\in}\mathbb{R}^{N\times d}$, and obtain local f_i^{local} and global f_i^{global} features separately through an MLP layer with a pipeline of $LayerNorm{\rightarrow}Linear(d, d/2){\rightarrow}GELU$:

$$f_i^{local} = \text{MLP}(x_i) \in \mathbb{R}^{N\times d/2}, \tag{1}$$

$$f_i^{global} = \text{AvgPool}(\text{MLP}(x_i), D) \in \mathbb{R}^{1\times d/2}, \tag{2}$$

where D is the keep/prune decision of the current tokens evaluated by Eq. (7). We then pass the combined feature $f_i{=}[f_i^{local}, f_i^{global}]{\in}\mathbb{R}^{N\times d}$ through a MLP pipeline of $Linear(d, d/2){\rightarrow}GELU{\rightarrow}Linear(d/2, d/4){\rightarrow}GELU{\rightarrow}Linear(d/4, 2)$ to produce a series of token score maps $\{t_i\}_{i=1}^{H}{\in}\mathbb{R}^{N\times 2}$, with t_i indicating the token score from each attention head:

$$t_i = \text{Softmax}(\text{MLP}(f_i)) \in \mathbb{R}^{N\times 2}, \tag{3}$$

where $N{\times}2$ represents the keep and prune probabilities of N number of tokens.

Head Attention Branch. We merge the individual score maps by the weights of each attention head to get the overall token score. As shown in Fig. 2, we add an attention-based branch along the selector backbone to synthesis the impor-

tance of each head:

$$\bar{X} = \text{AvgPool}(X) = Concat\{\frac{1}{C}\sum_{i=1}^{C} x_i\}_{j=1}^{H} \in \mathbb{R}^{N\times H}, \tag{4}$$

$$A = \text{Sigmoid}(\text{Linear}(\text{GeLU}(\text{Linear}(\bar{X})))) \in \mathbb{R}^{N\times H}, \tag{5}$$

where $\bar{X}$ is a head-wise statistic generated by shrinking X through its channel dimension C with global average pooling. In Eq. (5), the attention head score vector A is obtained by feeding $\bar{X}$ into the $Linear(H, H/2)\rightarrow GeLU\rightarrow Linear(H/2, H)\rightarrow Sigmoid$ pipeline to fully capture head-wise dependencies. The overall token score is calculated by adding the token scores from each individual attention head, multiplying by their individual head score $\{a_i\}_{i=1}^{H}\in\mathbb{R}^{N\times 1}$:

$$\tilde{T} = \frac{\sum_{i=1}^{H} t_i * a_i}{\sum_{i=1}^{H} a_i} \in \mathbb{R}^{N\times 2}, \tag{6}$$

where $\tilde{T}$ is the final token probability score. To make the token removing differentiable, we apply the Gumbel-Softmax technique to generate the token keep/prune decision during training:

$$D = \text{GumbelSoftmax}(\tilde{T}) \in \{0, 1\}^{N}. \tag{7}$$

Next, D passes on to the following layers until reaching the next token selector, where it will be updated by applying Hadamard product with the new token keep decision $D \odot D'$ during our hierarchical pruning scheme.

Self-attention matrices-based methods [46,89] usually require sorting and evaluating the importance of tokens by a Top-k operation, which is currently not supported in many frameworks for edge devices [66]. On the contrary, our selector generates binary matrices with the help of gumble softmax and FC layers to perform pruning instead of Top-k ordering. For hardware efficiency, our token selector mainly leverages the FC layers to reuse the GEMM hardware engine already built for the backbone ViT.

4.2 Token Packaging Technique

As discussed before, ViT is less accurate for evaluating token values in earlier blocks. Poor scoring may cause important tokens to be removed. Moreover, completely removing background (negative) tokens will weaken self-attention's ability to capture key information [92]. Instead of completely discarding tokens that are considered less informative, we apply a token packaging technique that integrates them into a package token. Assume there are Q less informative tokens $\hat{X} = \{n_i\}_{i=1}^{Q}$, $n_i \in \mathbb{R}^{C}$, along with their token scores $\hat{T} = \{m_i\}_{i=1}^{Q}$, $m_i \in \mathbb{R}^2$ These tokens are combined into one token by:

$$P = \frac{\sum_{i=1}^{Q} n_i \cdot m_i[0]}{\sum_{i=1}^{Q} m_i[0]} \in \mathbb{R}^{C}, \tag{8}$$

Table 2: Latency of one DeiT block on the Xilinx ZCU102 FPGA board.

Pruning Rate	0.0	0.1	0.2	0.3	0.4	0.5
DeiT-T Latency (ms)	0.689	0.630	0.587	0.509	0.468	0.424
DeiT-S Latency (ms)	2.107	1.891	1.710	1.503	1.315	1.121

where P is the package token; $m_i[0]$ is the probability of keeping the token. Token P will participate in the subsequent calculations along with the informative tokens, enabling the model to correct scoring mistakes. Our overall framework is efficient, with miniature computation cost (less than 1% of the total model GFLOPs). All the operations (MLP, Softmax, Pooling, Sigmoid, etc.) are well supported on edge platforms.

4.3 Latency-Aware Training Strategy

Our latency-aware training strategy includes two parts: (1) the training objective where we introduce the latency-aware sparsity loss to obtain the pruning rate of token constrained by the latency specifications of the target devices; (2) the layer-to-phase progressive training schedule by which we can determine the location of inserted selectors and their suitable pruning rates.

Latency-Sparsity Table. In order to bridge the inference of ViT model produced by SPViT to the actual latency bound of hardware operation, we measure the latency-sparsity table of the target device, shown in Table 2. Note that the computation amount of one selector is less than 1% of one ViT block and the specific latency can be disregarded.

Latency-Aware Sparsity Loss. Based on the relationship between the pruning rate and latency in Table 2, we introduce a latency-aware sparsity loss $\pounds_{ratio}$:

$$\text{Block\_lat}(\rho_i) = latency\_sparsity\_table(\rho_i), \tag{9}$$

$$\sum_{i=1}^{L} \text{Block\_lat}(\rho_i) \leq \text{LatencyLimit}, \tag{10}$$

$$\pounds_{ratio} = \sum_{i=1}^{L} (1 - \rho_i - \frac{1}{B} \sum_{b=1}^{B} \sum_{j=1}^{N} D_j^{i,b})^2, \tag{11}$$

where Eq. (9) is a look-up-table which aims to find the latency of one block Block_lat under the corresponding ratio ρ_i with Table 2. Eq. (10) guarantees that the inference latency of the model should be under the limit of target edge devices after token pruning. $LatencyLimit$ is the latency constraints of the target device. With i being the block index, ρ_i is the corresponding pruning rate. Through Eq. (9) and (10), we derive appropriate ρ_i and feed it to the final sparsity loss (11), where B is the training batch size, and $D_*^{i,*}$ (Eq. (7)) is token

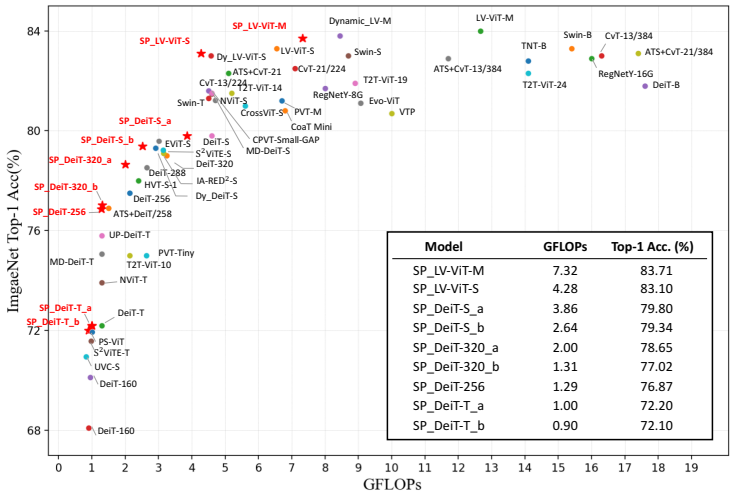
keep decision. In order to achieve per-image adaptive pruning, we set the average pruning rate of all images in one batch as the convergence target of the Eq. (11).
Training Objective. It includes the standard cross-entropy loss, soft distillation loss, and latency-aware sparsity loss. The former two are the same as the loss strategy used in DeiT [79].

$$\pounds = \pounds_{cls} + \lambda_{KL}\pounds_{KL} + \lambda_{distill}\pounds_{distill} + \lambda_{ratio}\pounds_{ratio}, \tag{12}$$

where we set λ_{KL}=0.5, $\lambda_{distill}$=0.5, λ_{ratio}=2 in all our experiments.
Layer-to-Phase Progressive Training Schedule. Based on [111], we assume that the final CLS token is strongly correlated with classification. And we use centered kernel alignment (CKA) similarity [42] to calculate the similarity of the token features in each block and the final CLS token. As shown in Fig. 4, the final CLS token feature is quite different from token features in earlier blocks. It shows that the representations in earlier blocks are encoded inadequately, which proves the difficulty of pruning tokens in the earlier blocks. Combined with this encoding pattern, we design a latency-aware progressive training strategy to find the optimal accuracy-pruning rate trade-offs and proper locations for token selectors. In a ViT, tokens can be more effectively encoded in later blocks. Hence, we adopt progressive training on the token selector from later blocks to earlier ones. Specifically, each time we insert a token selector, we train the current selector and finetune the other parts (backbone and other selectors) by increasing the pruning rate of the current block until accuracy decreases noticeably ($> 0.5\%$). We repeat the insertion until there is one selector for each block. Then if the adjacent selectors have a similar pruning rate (difference$<$ 8.5%), we combine them as one selection phase and solely keep the first selector of the phase. Finally, if the final computations are lower than the target latency of specific edge devices, we reduce the pruning rate of the first selector. This is because we observe that earlier blocks are more sensitive to pruning.

5 Experiments

Datasets and Implementation Details. Our experiments are conducted on ImageNet-1K [20] with different backbones including DeiT-T, DeiT-S [79]; LV-ViT-S, LV-ViT-M [40]; PiT-T, PiT-XS, PiT-S [35]; Swin-T, Swin-S [49]. The image resolution is 224×224. We follow most of the training settings as in DeiT and train all backbone models for 60 epochs. Through our layer-to-phase training, we observe that inserting three token pruning selectors is best for the computation-accuracy tradeoff. For DeiT-T/S, we insert the token selector after the 3rd, 6th, and 9th layers. For LV-ViT-S, we insert the token selector after the 4th, 8th, and 12th layers. For LV-ViT-M, we insert the token selector after the 5th, 10th, and 15th layers. For PiT-T/XS/S, we insert the token selector after the 1st, 5th, and 10th layers. For Swin-T/S, we insert the token selector after each patch merging layer at the 2nd, 3rd, and 4th stage. Our batch size is 256 for DeiT-T, DeiT-S, and LV-ViT-S; and 128 for LV-ViT-M, PiT-T, PiT-XS, and PiT-S. We set an initial learning rate to be 5e-4 for the soft pruning module and 5e-6 for



Model	GFLOPs	Top-1 Acc. (%)
SP_LV-ViT-M	7.32	83.71
SP_LV-ViT-S	4.28	83.10
SP_DeiT-S_a	3.86	79.80
SP_DeiT-S_b	2.64	79.34
SP_DeiT-320_a	2.00	78.65
SP_DeiT-320_b	1.31	77.02
SP_DeiT-256	1.29	76.87
SP_DeiT-T_a	1.00	72.20
SP_DeiT-T_b	0.90	72.10

Fig. 5: Computation (GFLOPs) and top-1 accuracy trade-offs on ImageNet. Our models can achieve better trade-offs compared to other pruned or scaled models.

the backbone. The final model has three token selectors. All models are trained on 8 NVIDIA A100-SXM4-40GB GPUs. The latency is measured on a Samsung Galaxy S20 cell phone that has Snapdragon 865 processor, which consists of an Octa-core Kryo 585 CPU.

5.1 Experimental Results

Main Results. We compare our method with several representative methods including DynamicViT [69], IA-RED^2 [64], RegNetY [67], CrossViT [7], VTP [112], ATS [28], CvT [85], PVT [83], T2T-ViT [104], UP-DeiT [95], PS-ViT [78], Evo-ViT [89], TNT [34], HVT [65], Swin [49], CoaT [88], CPVT [18], EViT [46], UVC [97], MD-DeiT [39],and S^{2}ViTE [12]. Fig. 5 demonstrates that our models achieve better accuracy-computation trade-offs compared to other pruned or scaled models. Our SPViT reduces the computation cost by 31%∼43% for various backbones with negligible 0.1%∼0.5% accuracy degradation, which outperforms existing methods on both accuracy and efficiency. On lightweight ViT, DeiT-T, the proposed SPViT still reduces GFLOPs by 31% with a negligible 0.1% decrease of accuracy (72.10% vs. 72.20%). To explore model scaling on ViT, we train more DeiT models with the embedding dimension of 160/256/288/320 as our baselines. On DeiT-T and DeiT-S under the same or similar GFLOPs, the accuracy improvement of SPViT over DeiT-160 is 4% (72.1% vs. 68.1% with $\sim$ 0.9 GFLOPs), 4.67% (76.87% vs. 72.20% with $\sim$1.3 GFLOPs) of SPViT-256 over DeiT-T-192, 4.82% (77.02% vs. 72.20% with $\sim$1.3 GFLOPs) of SPViT-320 over DeiT-T-192, and 0.81% (79.34% vs. 78.53% with $\sim$2.65 GFLOPs) of SPViT over DeiT-S-288. Additionally, our method can prune up to 23.1% on DeiT-T and 16.1% on DeiT-S without any accuracy degradation.

Table 3: Evaluation results on Hierarchical Architectures with SPViT.

Model	GFLOPs	Top1 Acc (%)
Swin-S	8.70	83.20
SPViT (Ours)	6.35 (26.4% ↓)	**82.71** (0.49% ↓)
Swin-T	4.50	81.20
SPViT (Ours)	3.47 (23.0% ↓)	**80.70** (0.50% ↓)
PiT-S	2.90	80.90
SPViT (Ours)	2.22 (23.3% ↓)	**80.32** (0.58% ↓)
PiT-XS	1.40	78.10
SPViT (Ours)	1.13 (18.7% ↓)	**77.86** (0.24% ↓)

Table 4: Evaluation results on Samsung Galaxy S20 with Snapdragon 865 processor and Xilinx ZCU102 FPGA board.

Model	Method	Top-1 Acc. (%)	Latency (ms)
Samsung Galaxy S20			
DeiT-T	Baseline	72.20	44
	SPViT (Ours)	**72.10**	**26**
DeiT-S	Baseline	79.80	113
	SPViT (Ours)	**79.34**	**60**
Xilinx ZCU102 FPGA			
DeiT-T	Baseline	72.20	8.81
	SPViT (Ours)	**72.10**	**5.60**
DeiT-S	Baseline	79.80	22.31
	SPViT (Ours)	**79.34**	**13.23**

Results on Hierarchical Architectures. We also perform SPViT on lightweight hierarchical ViTs: Swin-Transformer and PiT, and present the results in Table 3. Our SPViT reduces the computation cost by 23%∼27% for Swin with a slight accuracy degradation of 0.4%∼0.5%, and by 18%∼24% for PiT with a degradation of 0.2%∼0.6%. Even though Swin has scaled down the computation complexity to $O(N)$ through window-based self-attention, and PiT is already a lightweight ViT model, we still can achieve a fair amount of compression while keeping the accuracy intact.

5.2 Deployment on Edge Devices

To evaluate the hardware performance, we implement a framework that runs the ViT model on edge devices. The evaluation is conducted on a Samsung Galaxy S20 cell phone that has a Snapdragon 865 processor, which consists of an Octa-core Kryo 585 CPU carrying high performance with good power efficiency. We use all eight cores on mobile CPUs. We report the average latency of over 100 inferences. As shown in Fig. 1, our method outperforms existing pruning methods on both latency and accuracy. The deficiencies of other methods mainly lie in three categories: limited pruning capability (low pruning rate) [64], non-optimal pruning dimension (number of heads) [12], and less efficient operators (e.g., Argsort.) [69]. As shown in Table 4, on the one hand, our models can outperform lightweight models such as DeiT-T by up to 4.8% under similar latency. On the other hand, we are able to reduce the latency of larger models such as DeiT-S by up to 47% (60ms vs. 113ms) with only 0.46% decrease of accuracy. Especially, for DeiT-T, we achieve 26 ms per inference on mobile CPUs, which meets the real-time requirement. As far as we know, this is the first demonstration of ViT inference over 30 fps on edge devices.

Additionally, SPViT is evaluated on an embedded FPGA platform, Xilinx ZCU102. To maintain the model accuracy on hardware, 16-bit fixed-point precision is adopted to represent all the model parameters and activation data. The comparison results with baseline models are shown in Table 4. In addition to the total latency, the average latency of the multi-head attention and MLP modules

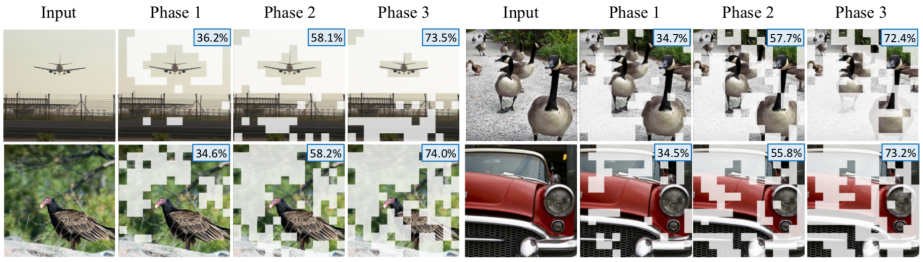


Fig. 6: Visualization of each pruning phase. In the 1st phase, the selector removes part of the background. In the 2nd phase, it targets the object of interest closely. In the 3rd phase, it localizes the informative features of the objects. The top right corner of each image shows the pruning rate after each phase.

Table 5: Token selector number/location evaluation on DeiT-S.

Location	Params (M)	GFLOPs	Top-1 Acc. (%)
3-6-9	22.13	**2.65**	**79.34**
1-6-9	22.13	2.70	76.10
3-6-11	22.13	2.72	78.76
6-9	22.10	2.71	78.53
3-5-7-9	22.16	2.66	79.34

Table 6: Comparison of different pruning methods.

Model	Method	GFLOPs	Top-1 Acc. (%)
DeiT-T	Random	0.90	69.87
	Structure	0.90	70.32
	Token selector	**0.90**	**72.10**
DeiT-S	Random	2.64	77.25
	Structure	2.64	77.86
	Token selector	**2.64**	**79.34**

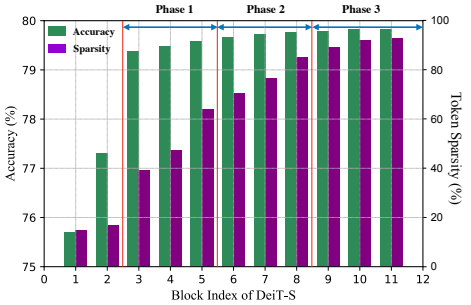


Fig. 7: The accuracy and the token sparsity distribution after the Layer-to-Phase Progressive Training. We do the insertion behind the $Block_{index}$. Our final phase plan is demonstrated above.

in each model is listed. Compared with the baseline, DeiT-T and DeiT-S, SPViT could achieve 1.57× and 1.69× acceleration in the total latency, respectively.

5.3 Token Pruning Visualization

We further visualize the hierarchical token reduction process of SPViT within Fig. 6. We show the input images along with their sparsification results after each phase. The masked regions represent the tokens that have been soft pruned. Our SPViT can gradually drop less informative tokens and preserve the tokens that contain representative regions with an adaptive pruning rate for each image.

5.4 Ablation Analysis

Token Selector Number and Location. After progressive training (each selector is fine-tuned by 25 epochs), we can get the pruning rate of each block as

shown in Fig. 7. Based on the trend of the figure, we can divide the evolution of the pruning rate into 2 phases, 3 phases, and 4 phases. We keep the appropriate selectors accordingly and re-finetuning the whole model. In Table 5, the 3-6-9 division style has the highest accuracy and the lowest computation cost, just like 3-5-7-9. According to the test on Samsung Galaxy S20, each selector and corresponding package token will introduce a delay of 1.67 ms, so we choose 3-6-9 as the best. For another 3-phase style, 1-6-9, the accuracy and computation cost are both not ideal. This shows that due to insufficient encoding, it is difficult to perform token pruning in the earlier blocks of ViTs. Meanwhile, for the 3-6-11 style, both the accuracy and computation cost are slightly inferior to the 3-6-9 style. The possible reason is the pruning rate of the second phase should be smaller than the third phase and the coverage of the second phase is too wide. As a result, there is still a lot of redundancy in the tokens of the third phase, restricting the accuracy and computation efficiency of the model at the same time. Furthermore, because of a similar reason, the 2-phase style, 3-6, cannot achieve a better trade-off between accuracy and the computation cost.

Comparison of Different Pruning Methods To further prove the effectiveness of our score-based dynamic token pruning method, we compare with some general pruning methods: random pruning and structure pruning. For random pruning, we randomly remove the input token, neglecting the token importance. For structure pruning, we prune the input feature map by dimension, which will impair every token. Results are shown in Table 6. Under the same computational complexities (0.9 GFLOPs for DeiT-T and 2.64 GFLOPs for DeiT-S), our proposed method achieves the best accuracy.

5.5 Limitations

For the algorithm design, it might be more effective to combine our framework with the weight pruning strategy for larger ViTs. For the hardware deployment, large amounts of data movement bring much pressure to the memory due to multiple blocks and many intermediate results, which will be optimized in our further work.

6 Conclusion

In this paper, we propose a dynamic, latency-aware soft token pruning framework called SPViT. Our attention-based multi-head token selector and token packaging technique, along with the latency-aware training strategy can well balance the tradeoff between accuracy and specific hardware constraints. We deploy our model on mobile and FPGA, which both meet the real-time requirement.

Acknowledgments. The research reported here was funded in whole or in part by the Army Research Office/Army Research Laboratory via grant W911-NF-20-1-0167 to Northeastern University. Any errors and opinions are not those of the Army Research Office or Department of Defense and are attributable solely to the author(s). This research is also partially supported by National Science Foundation CCF-1919117 and CMMI-2125326.

References

1. Amini, A., Periyasamy, A.S., Behnke, S.: T6d-direct: Transformers for multi-object 6d pose direct regression. arXiv preprint arXiv:2109.10948 (2021)
2. Bao, H., Dong, L., Piao, S., Wei, F.: BEit: BERT pre-training of image transformers. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=p-BhZSz59o4
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
4. Chang, S.E., Li, Y., Sun, M., Shi, R., So, H.K.H., Qian, X., Wang, Y., Lin, X.: Mix and match: A novel fpga-centric deep neural network quantization framework. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). pp. 208–220. IEEE (2021)
5. Chefer, H., Gur, S., Wolf, L.: Transformer interpretability beyond attention visualization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 782–791 (2021)
6. Chen, B., Li, P., Li, B., Li, C., Bai, L., Lin, C., Sun, M., Yan, J., Ouyang, W.: Psvit: Better vision transformer via token pooling and attention sharing. arXiv preprint arXiv:2108.03428 (2021)
7. Chen, C.F.R., Fan, Q., Panda, R.: Crossvit: Cross-attention multi-scale vision transformer for image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 357–366 (2021)
8. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., Gao, W.: Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12299–12310 (2021)
9. Chen, M., Peng, H., Fu, J., Ling, H.: Autoformer: Searching transformers for visual recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12270–12280 (2021)
10. Chen, P., Chen, Y., Liu, S., Yang, M., Jia, J.: Exploring and improving mobile level vision transformers. arXiv preprint arXiv:2108.13015 (2021)
11. Chen, T., Chen, X., Ma, X., Wang, Y., Wang, Z.: Coarsening the granularity: Towards structurally sparse lottery tickets. In: Proceedings of the International Conference on Machine Learning (ICML) (2022)
12. Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., Wang, Z.: Chasing sparsity in vision transformers: An end-to-end exploration. In: Advances in Neural Information Processing Systems (2021)
13. Chen, T., Saxena, S., Li, L., Fleet, D.J., Hinton, G.: Pix2seq: A language modeling framework for object detection. arXiv preprint arXiv:2109.10852 (2021)
14. Chen, X., Hsieh, C.J., Gong, B.: When vision transformers outperform resnets without pre-training or strong data augmentations. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=LtKcMgGOeLt
15. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8126–8135 (2021)
16. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021), https://openreview.net/forum?id=0lz69oI5iZP

17. Chu, C., Wang, Y., Zhao, Y., Ma, X., Ye, S., Hong, Y., Liang, X., Han, Y., Jiang, L.: Pim-prune: Fine-grain dcnn pruning for crossbar-based process-in-memory architecture. In: 2020 57th ACM/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2020)
18. Chu, X., Tian, Z., Zhang, B., Wang, X., Wei, X., Xia, H., Shen, C.: Conditional positional encodings for vision transformers. arXiv preprint arXiv:2102.10882 (2021)
19. Dai, Z., Cai, B., Lin, Y., Chen, J.: Up-detr: Unsupervised pre-training for object detection with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1601–1610 (2021)
20. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
21. Deng, J., Yang, Z., Chen, T., Zhou, W., Li, H.: Transvg: End-to-end visual grounding with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1769–1779 (October 2021)
22. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy
23. El-Nouby, A., Neverova, N., Laptev, I., Jégou, H.: Training vision transformers for image retrieval. arXiv preprint arXiv:2102.05644 (2021)
24. El-Nouby, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., Jegou, H.: XCit: Cross-covariance image transformers. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021), https://openreview.net/forum?id=kzPtpIpF8o
25. Fang, H., Mei, Z., Shrestha, A., Zhao, Z., Li, Y., Qiu, Q.: Encoding, model, and architecture: Systematic optimization for spiking neural network in fpgas. In: 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD). pp. 1–9. IEEE (2020)
26. Fang, H., Shrestha, A., Zhao, Z., Qiu, Q.: Exploiting neuron and synapse filter dynamics in spatial temporal learning of deep spiking neural network. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. IJCAI'20 (2021)
27. Fang, H., Taylor, B., Li, Z., Mei, Z., Li, H.H., Qiu, Q.: Neuromorphic algorithm-hardware codesign for temporal pattern learning. In: 2021 58th ACM/IEEE Design Automation Conference (DAC). pp. 361–366. IEEE (2021)
28. Fayyaz, M., Kouhpayegani, S., Jafari, F., Sommerlade, E., Joze, H., Pirsiavash, H., Gall, J.: Ats: Adaptive token sampling for efficient vision transformers. arXiv preprint arXiv:2111.15667 (2021)
29. Gao, P., Lu, J., Li, H., Mottaghi, R., Kembhavi, A.: Container: Context aggregation network. arXiv preprint arXiv:2106.01401 (2021)
30. Gong, Y., Zhan, Z., Li, Z., Niu, W., Ma, X., Wang, W., Ren, B., Ding, C., Lin, X., Xu, X., et al.: A privacy-preserving-oriented dnn pruning and mobile acceleration framework. In: Proceedings of the 2020 on Great Lakes Symposium on VLSI. pp. 119–124 (2020)
31. Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jegou, H., Douze, M.: Levit: A vision transformer in convnet's clothing for faster inference. In: Pro-

ceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12259–12269 (October 2021)
32. Guo, C., Hsueh, B.Y., Leng, J., Qiu, Y., Guan, Y., Wang, Z., Jia, X., Li, X., Guo, M., Zhu, Y.: Accelerating sparse dnn models without hardware-support via tile-wise sparsity. In: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1–15. IEEE (2020)
33. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (Apr 2021). https://doi.org/10.1007/s41095-021-0229-5, http://dx.doi.org/10.1007/s41095-021-0229-5
34. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. In: Advances in Neural Information Processing Systems (2021)
35. Heo, B., Yun, S., Han, D., Chun, S., Choe, J., Oh, S.J.: Rethinking spatial dimensions of vision transformers. In: International Conference on Computer Vision (ICCV) (2021)
36. Hou, Z., Qin, M., Sun, F., Ma, X., Yuan, K., Xu, Y., Chen, Y.K., Jin, R., Xie, Y., Kung, S.Y.: Chex: Channel exploration for cnn model compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12287–12298 (2022)
37. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
38. Hudson, D.A., Zitnick, C.L.: Generative adversarial transformers. Proceedings of the 38th International Conference on Machine Learning, ICML 2021 (2021)
39. Jia, D., Han, K., Wang, Y., Tang, Y., Guo, J., Zhang, C., Tao, D.: Efficient vision transformers via fine-grained manifold distillation. arXiv preprint arXiv:2107.01378 (2021)
40. Jiang, Z., Hou, Q., Yuan, L., Zhou, D., Shi, Y., Jin, X., Wang, A., Feng, J.: All tokens matter: Token labeling for training better vision transformers. arXiv preprint arXiv:2104.10858 (2021)
41. Kim, B., Lee, J., Kang, J., Kim, E.S., Kim, H.J.: Hotr: End-to-end human-object interaction detection with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 74–83 (2021)
42. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: International Conference on Machine Learning. pp. 3519–3529. PMLR (2019)
43. Li, B., Kong, Z., Zhang, T., Li, J., Li, Z., Liu, H., Ding, C.: Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 3187–3199 (2020)
44. Li, Y., Fang, H., Li, M., Ma, Y., Qiu, Q.: Neural network pruning and fast training for drl-based uav trajectory planning. In: 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC). pp. 574–579. IEEE (2022)
45. Li, Z., Liu, X., Drenkow, N., Ding, A., Creighton, F.X., Taylor, R.H., Unberath, M.: Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6197–6206 (2021)
46. Liang, Y., GE, C., Tong, Z., Song, Y., Wang, J., Xie, P.: EVit: Expediting vision transformers via token reorganizations. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=BjyvwnXXVn_

47. Liu, N., Yuan, G., Che, Z., Shen, X., Ma, X., Jin, Q., Ren, J., Tang, J., Liu, S., Wang, Y.: Lottery ticket preserves weight correlation: Is it desirable or not? In: International Conference on Machine Learning (ICML). pp. 7011–7020. PMLR (2021)
48. Liu, Y., Sangineto, E., Bi, W., Sebe, N., Lepri, B., De Nadai, M.: Efficient training of visual transformers with small-size datasets. arXiv preprint arXiv:2106.03746 (2021)
49. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. International Conference on Computer Vision (ICCV) (2021)
50. Lu, Z., Liu, H., Li, J., Zhang, L.: Efficient transformer for single image super-resolution. arXiv preprint arXiv:2108.11084 (2021)
51. Ma, X., Guo, F.M., Niu, W., Lin, X., Tang, J., Ma, K., Ren, B., Wang, Y.: PCONV: The missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). vol. 34, pp. 5117–5124 (2020)
52. Ma, X., Lin, S., Ye, S., He, Z., Zhang, L., Yuan, G., Tan, S.H., Li, Z., Fan, D., Qian, X., et al.: Non-structured dnn weight pruning–is it beneficial in any platform? IEEE Transactions on Neural Networks and Learning Systems (TNNLS) (2021)
53. Ma, X., Niu, W., Zhang, T., Liu, S., Lin, S., Li, H., Wen, W., Chen, X., Tang, J., Ma, K., et al.: An image enhancing pattern-based sparsity for real-time inference on mobile devices. In: Proceedings of the European conference on computer vision (ECCV). pp. 629–645. Springer (2020)
54. Ma, X., Qin, M., Sun, F., Hou, Z., Yuan, K., Xu, Y., Wang, Y., Chen, Y.K., Jin, R., Xie, Y.: Effective model sparsification by scheduled grow-and-prune methods. In: Proceedings of the International Conference on Learning Representations (ICLR) (2021)
55. Ma, X., Yuan, G., Li, Z., Gong, Y., Zhang, T., Niu, W., Zhan, Z., Zhao, P., Liu, N., Tang, J., et al.: Blcr: Towards real-time dnn execution with block-based reweighted pruning. In: International Symposium on Quality Electronic Design (ISQED). pp. 1–8. IEEE (2022)
56. Ma, X., Yuan, G., Lin, S., Ding, C., Yu, F., Liu, T., Wen, W., Chen, X., Wang, Y.: Tiny but accurate: A pruned, quantized and optimized memristor crossbar framework for ultra efficient dnn implementation. In: 2020 25th Asia and South Pacific design automation conference (ASP-DAC). pp. 301–306. IEEE (2020)
57. Ma, X., Yuan, G., Shen, X., Chen, T., Chen, X., Chen, X., Liu, N., Qin, M., Liu, S., Wang, Z., et al.: Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? Advances in Neural Information Processing Systems (NeurIPS) **34** (2021)
58. Mao, M., Zhang, R., Zheng, H., Gao, P., Ma, T., Peng, Y., Ding, E., Han, S.: Dual-stream network for visual recognition. In: Advances in Neural Information Processing Systems (2021)
59. Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C.: Trackformer: Multi-object tracking with transformers. arXiv preprint arXiv:2101.02702 (2021)
60. Misra, I., Girdhar, R., Joulin, A.: An End-to-End Transformer Model for 3D Object Detection. In: ICCV (2021)
61. Niu, W., Kong, Z., Yuan, G., Jiang, W., Guan, J., Ding, C., Zhao, P., Liu, S., Ren, B., Wang, Y.: A compression-compilation framework for on-mobile real-time bert applications. arXiv preprint arXiv:2106.00526 (2021)

62. Niu, W., Li, Z., Ma, X., Dong, P., Zhou, G., Qian, X., Lin, X., Wang, Y., Ren, B.: Grim: A general, real-time deep learning inference framework for mobile devices based on fine-grained structured weight sparsity. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2021)
63. Niu, W., Ma, X., Lin, S., Wang, S., Qian, X., Lin, X., Wang, Y., Ren, B.: Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). pp. 907–922 (2020)
64. Pan, B., Jiang, Y., Panda, R., Wang, Z., Feris, R., Oliva, A.: Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. In: Advances in Neural Information Processing Systems (2021)
65. Pan, Z., Zhuang, B., Liu, J., He, H., Cai, J.: Scalable vision transformers with hierarchical pooling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 377–386 (2021)
66. Prillo, S., Eisenschlos, J.: Softsort: A continuous relaxation for the argsort operator. In: International Conference on Machine Learning. pp. 7793–7802. PMLR (2020)
67. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10428–10436 (2020)
68. Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., Dosovitskiy, A.: Do vision transformers see like convolutional neural networks? arXiv preprint arXiv:2108.08810 (2021)
69. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. In: Advances in Neural Information Processing Systems (2021)
70. Ren, A., Zhang, T., Ye, S., Li, J., Xu, W., Qian, X., Lin, X., Wang, Y.: Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 925–938 (2019)
71. Renggli, C., Pinto, A.S., Houlsby, N., Mustafa, B., Puigcerver, J., Riquelme, C.: Learning to merge tokens in vision transformers. arXiv preprint arXiv:2202.12015 (2022)
72. Rumi, M.A., Ma, X., Wang, Y., Jiang, P.: Accelerating sparse cnn inference on gpus with performance-aware weight pruning. In: Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques (PACT). pp. 267–278 (2020)
73. Ryoo, M.S., Piergiovanni, A., Arnab, A., Dehghani, M., Angelova, A.: Tokenlearner: What can 8 learned tokens do for images and videos? In: Advances in Neural Information Processing Systems (2021)
74. Sanh, V., Wolf, T., Rush, A.M.: Movement pruning: Adaptive sparsity by fine-tuning. arXiv preprint arXiv:2005.07683 (2020)
75. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16519–16529 (2021)
76. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021)

77. Tan, Z., Song, J., Ma, X., Tan, S.H., Chen, H., Miao, Y., Wu, Y., Ye, S., Wang, Y., Li, D., et al.: Pcnn: Pattern-based fine-grained regular pruning towards optimizing cnn accelerators. In: 2020 57th ACM/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2020)
78. Tang, Y., Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., Tao, D.: Patch slimming for efficient vision transformers (2021)
79. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., J'egou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
80. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
81. Wang, H., Zhang, Z., Han, S.: Spatten: Efficient sparse attention architecture with cascade token and head pruning. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). pp. 97–110. IEEE (2021)
82. Wang, P., Wang, X., Wang, F., Lin, M., Chang, S., Xie, W., Li, H., Jin, R.: Kvt: k-nn attention for boosting vision transformers. arXiv preprint arXiv:2106.00515 (2021)
83. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: IEEE ICCV (2021)
84. Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., Vajda, P.: Visual transformers: Token-based image representation and processing for computer vision. arXiv preprint arXiv:2006.03677 (2020)
85. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: Cvt: Introducing convolutions to vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 22–31 (October 2021)
86. Wu, K., Peng, H., Chen, M., Fu, J., Chao, H.: Rethinking and improving relative position encoding for vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10033–10041 (2021)
87. Xu, C., Zhai, B., Wu, B., Li, T., Zhan, W., Vajda, P., Keutzer, K., Tomizuka, M.: You only group once: Efficient point-cloud processing with token representation and relation inference module. arXiv preprint arXiv:2103.09975 (2021)
88. Xu, W., Xu, Y., Chang, T., Tu, Z.: Co-scale conv-attentional image transformers. arXiv preprint arXiv:2104.06399 (2021)
89. Xu, Y., Zhang, Z., Zhang, M., Sheng, K., Li, K., Dong, W., Zhang, L., Xu, C., Sun, X.: Evo-vit: Slow-fast token evolution for dynamic vision transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence (2022)
90. Xue, F., Wang, Q., Guo, G.: Transfer: Learning relation-aware facial expression representations with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3601–3610 (2021)
91. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. arXiv preprint arXiv:2103.17154 (2021)
92. Yang, C., Wu, Z., Zhou, B., Lin, S.: Instance localization for self-supervised detection pretraining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3987–3996 (2021)
93. Yang, F., Yang, H., Fu, J., Lu, H., Guo, B.: Learning texture transformer network for image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5791–5800 (2020)
94. Yang, G., Tang, H., Ding, M., Sebe, N., Ricci, E.: Transformer-based attention networks for continuous pixel-wise prediction. In: ICCV (2021)

95. Yu, H., Wu, J.: A unified pruning framework for vision transformers. arXiv preprint arXiv:2111.15127 (2021)
96. Yu, Q., Xia, Y., Bai, Y., Lu, Y., Yuille, A., Shen, W.: Glance-and-gaze vision transformer. In: Advances in Neural Information Processing Systems (2021)
97. Yu, S., Chen, T., Shen, J., Yuan, H., Tan, J., Yang, S., Liu, J., Wang, Z.: Unified visual transformer compression. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=9jsZiUgkCZP
98. Yuan, G., Behnam, P., Cai, Y., Shafiee, A., Fu, J., Liao, Z., Li, Z., Ma, X., Deng, J., Wang, J., et al.: Tinyadc: Peripheral circuit-aware weight pruning framework for mixed-signal dnn accelerators. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 926–931. IEEE (2021)
99. Yuan, G., Liao, Z., Ma, X., Cai, Y., Kong, Z., Shen, X., Fu, J., Li, Z., Zhang, C., Peng, H., et al.: Improving dnn fault tolerance using weight pruning and differential crossbar mapping for reram-based edge ai. In: 2021 22nd International Symposium on Quality Electronic Design (ISQED). pp. 135–141. IEEE (2021)
100. Yuan, G., Ma, X., Ding, C., Lin, S., Zhang, T., Jalali, Z.S., Zhao, Y., Jiang, L., Soundarajan, S., Wang, Y.: An ultra-efficient memristor-based dnn framework with structured weight pruning and quantization using admm. In: 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). pp. 1–6. IEEE (2019)
101. Yuan, G., Ma, X., Niu, W., Li, Z., Kong, Z., Liu, N., Gong, Y., Zhan, Z., He, C., Jin, Q., et al.: Mest: Accurate and fast memory-economic sparse training framework on the edge. Advances in Neural Information Processing Systems (NeurIPS) **34** (2021)
102. Yuan, K., Guo, S., Liu, Z., Zhou, A., Yu, F., Wu, W.: Incorporating convolution designs into visual transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 579–588 (October 2021)
103. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.H., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 558–567 (October 2021)
104. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986 (2021)
105. Yue, X., Sun, S., Kuang, Z., Wei, M., Torr, P.H., Zhang, W., Lin, D.: Vision transformer with progressive sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 387–396 (October 2021)
106. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. arXiv preprint arXiv:2106.04560 (2021)
107. Zhang, T., Ma, X., Zhan, Z., Zhou, S., Ding, C., Fardad, M., Wang, Y.: A unified dnn weight pruning framework using reweighted optimization methods. In: 2021 58th ACM/IEEE Design Automation Conference (DAC). pp. 493–498. IEEE (2021)
108. Zhang, T., Ye, S., Feng, X., Ma, X., Zhang, K., Li, Z., Tang, J., Liu, S., Lin, X., Liu, Y., et al.: Structadmm: Achieving ultrahigh efficiency in structured pruning for dnns. IEEE Transactions on Neural Networks and Learning Systems (TNNLS) (2021)
109. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021)

110. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6881–6890 (2021)
111. Zhou, D., Shi, Y., Kang, B., Yu, W., Jiang, Z., Li, Y., Jin, X., Hou, Q., Feng, J.: Refiner: Refining self-attention for vision transformers (2021)
112. Zhu, M., Han, K., Tang, Y., Wang, Y.: Visual transformer pruning. In: KDD 2021 Workshop on Model Mining (2021)