Sparsifiner: Learning Sparse Instance-Dependent Attention for Efficient Vision Transformers

Cong Wei[1,3*] Brendan Duke[1,3*] Ruowei Jiang[3] Parham Aarabi[1,3] Graham W. Taylor[2,4] Florian Shkurti[1,4]

[1]University of Toronto [2]University of Guelph [3]Modiface, Inc. [4]Vector Institute

Abstract

Vision Transformers (ViT) have shown competitive advantages in terms of performance compared to convolutional neural networks (CNNs), though they often come with high computational costs. To this end, previous methods explore different attention patterns by limiting a fixed number of spatially nearby tokens to accelerate the ViT's multi-head self-attention (MHSA) operations. However, such structured attention patterns limit the token-to-token connections to their spatial relevance, which disregards learned semantic connections from a full attention mask. In this work, we propose an approach to learn instance-dependent attention patterns, by devising a lightweight connectivity predictor module that estimates the connectivity score of each pair of tokens. Intuitively, two tokens have high connectivity scores if the features are considered relevant either spatially or semantically. As each token only attends to a small number of other tokens, the binarized connectivity masks are often very sparse by nature and therefore provide the opportunity to reduce network FLOPs via sparse computations. Equipped with the learned unstructured attention pattern, sparse attention ViT (Sparsifiner) produces a superior Pareto frontier between FLOPs and top-1 accuracy on ImageNet compared to token sparsity. Our method reduces 48% $\sim$ 69% FLOPs of MHSA while the accuracy drop is within 0.4%. We also show that combining attention and token sparsity reduces ViT FLOPs by over 60%.

1. Introduction

Vision Transformers (ViTs) [15] have emerged as a dominant model for fundamental vision tasks, such as image classification [15], object detection [3], and semantic segmentation [6, 7]. However, scaling ViTs to a large number of tokens is challenging due to the quadratic computational complexity of multi-head self-attention (MHSA) [37]. This is particularly disadvantageous for large-scale vision tasks because processing high-resolution and high-dimensionality inputs is desirable. For example, input modalities such as video frames and 3D point clouds have a large number of tokens even for basic use cases. New algorithms are needed to continue to scale ViTs to larger, more complex vision tasks.

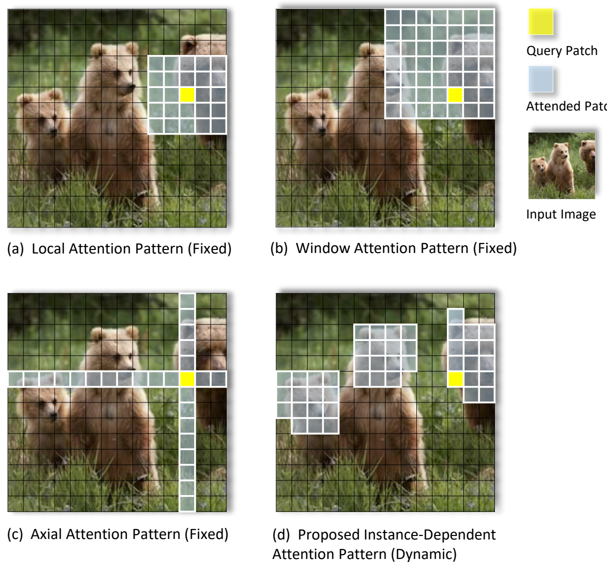

Query Patch
Attended Patch
Input Image
(a) Local Attention Pattern (Fixed)
(b) Window Attention Pattern (Fixed)
(c) Axial Attention Pattern (Fixed)
(d) Proposed Instance-Dependent Attention Pattern (Dynamic)

Figure 1. Comparison of Sparsifiner and fixed attention patterns. Twins [10] (a), Swin [24] (b), and Axial [18] (c) address quadratic MHSA complexity using fixed attention patterns, which does not consider the instance-dependent nature of semantic information in images. To address this, we propose Sparsifiner (d): an efficient module for sparse instance-dependent attention pattern prediction.

Prior works have largely taken two approaches to improve the computational efficiency of ViTs: token pruning and using fixed sparse attention patterns in MHSA. *Token pruning* methods [29] reduce the number of tokens by a fixed ratio called the keep rate, but accuracy degrades quickly when pruning early layers in the network [17, 32, 33]. For example, introducing token pruning into shallower layers of EViT [17] causes a significant 3.16% top-1 accuracy drop on ImageNet [14]. This issue is due to the restric-

*Equal contribution.

tion of pruning an entire token, which amounts to pruning an entire row and column of the attention matrix at once. One way to alleviate this is to prune connectivity between individual tokens in the attention matrix. Existing methods that take this attention matrix connectivity-pruning approach use *fixed sparse attention patterns* [8]. For example, local and strided fixed attention patterns are used [8, 16], in combination with randomly-initialized connectivities [42]. However, such fixed attention patterns limit the capacity of the self-attention connections to a fixed subset of tokens (Fig. 1). The fixed nature of these attention patterns is less effective compared to the direct communication between tokens in full self-attention. For example, Swin transformer [23, 24] has a limited the receptive field at shallower layers and needs many layers to model long-range dependencies. And BigBird [42] needs to combine multiple fixed attention patterns to achieve good performance. Rather, it is desirable to design sparse attention algorithms that mimic full self attention's instance-dependent nature [37], thereby capturing the variable distribution of semantic information in the input image content.

To address these challenges, we propose a method called Sparsifiner that learns to compute sparse connectivity patterns over attention that are both *instance-dependent* and *unstructured*. The instance-dependent nature of the attention pattern allows each token to use its limited attention budget of nonzero elements more efficiently compared to fixed sparse attention patterns. For example, in attention heads that attend to semantic rather than positional content [37, 38], tokens containing similar semantic information should be considered to have high connectivity scores despite their spatial distance. Similarly, nearby tokens with irrelevant semantic relations should have lower connectivity scores despite their spatial proximity. Furthermore, Sparsifiner improves attention pattern flexibility compared to token pruning by pruning individual connectivities instead of entire rows and columns of the attention matrix. This allows Sparsifiner to reduce FLOPs in the early layers of the network without incurring significant top-1 accuracy degradation (§4). By pruning individual connectivities dependent on image content, Sparsifiner generalizes prior approaches to sparsifying MHSA in ViTs, and in doing so, produces a favourable trade-off between accuracy and FLOPs.

Our contributions are the following:

- We propose a novel efficient algorithm called Sparsifiner to predict instance-dependent sparse attention patterns using low-rank connectivity patterns. Our investigation into instance-dependent unstructured sparsity is to the best of our knowledge novel in the context of ViTs.
- We show that such learned unstructured attention sparsity produces a superior Pareto-optimal tradeoff between FLOPs and top-1 accuracy on ImageNet compared to token sparsity. Furthermore, we show that Sparsifiner is complementary to token sparsity methods, and the two approaches can be combined to achieve superior performance-accuracy tradeoffs.
- We propose a knowledge distillation-based approach for training Sparsifiner from pretrained ViTs using a small number of training epochs.

2. Related Work

Efficient Attention — Developing an efficient attention mechanism for high-resolution image encoding is the focus of this work. Efficient attention mechanisms have been widely studied in NLP tasks to model long sequences. They can be categorized as follows: **Matrix factorization** such as Linformer [39] use a low-rank projection to linearize the attention operation, Monarch [12] transforms the attention matrix into factors of block-diagonal matrices and FlashAttention [13] further improves block sparse attention with tiling for superior GPU memory locality. **Kernelization**, including Performer [9], Linear Transformers [20], and Random Feature Attention [26] use kernels to avoid explicitly computing the attention matrix. **Sparse attention with fixed attention patterns** [8, 10, 18, 25, 27]. This type of technique sparsifies the attention matrix by limiting the field of view to predefined patterns such as local and strided windows. **Similarity and clustering** including Routing Transformer [31], Reformer [21], and Sinkhorn Transformer [35]. These models measure token relevance by sorting or clustering and then assign tokens to buckets for within-bucket attention. **Neural memory** such as Set Transformer [22], Compressive Transformer [28], and Longformer [1]. These use extra global tokens that gather long-range information as a model memory.

Vision Transformers — Recent progress has demonstrated that variants of Transformers [37] can also be competitive alternatives to CNNs and achieve promising results on different vision tasks. In addition to image classification, Transformers have also been applied to various vision tasks, including object detection [4, 11, 46, 48], image generation [5, 25], and video processing [44, 47]. Vision Transformer (ViT) [15] splits images as small patches and treats the patches as the input word tokens. ViT shows better performance than CNN-type models with sufficient extensive training data. DeiT [36] incorporates knowledge distillation techniques into ViT training so that we can train a competitive Transformer using only ImageNet-1k [14]. LV-ViT [19] further improves the performance of ViT by introducing a new training objective named token labelling. Most of these methods have quadratic complexity of self-attention with respect to the input image size.

Efficient Vision Transformers — There is a thrust to model long sequences of image patches at much higher resolutions. Recent works such as Pyramid Vision Transformer (PVT) [40], Swin-Transformer [24], T2T-ViT [41], and Vision Longformer (ViL) [45] apply transformer layers on different resolution scales by stacking a pyramid of ViTs to form a multi-scale architecture. To achieve linear complexity, Swin-Transformer [24] uses shifted local window attention. Vision Longformer [45] adapts the local attention pattern with the global memory tokens from Longformer [1]. TimeSformer [2] applies multiple attentions, each along a single axis of the input video. Those methods all leverage fixed, predefined attention patterns to reduce the quadratic cost. In contrast, our method generates sparse dynamic attention patterns based on the input content. Another group of works reduce the number of tokens by pruning [17,29,34], or merging tokens [30,32,43]. Recent work, DynamicViT [29] and EViT [17] study unstructured token sparsification by gradually dropping tokens in the inference of ViTs [15]. However, quadratic attention cost remains in early layers where input tokens cannot be largely sparsified. Our method instead prunes connectivities at every layer, allowing complexity savings at early layers.

3. Method

Our proposed method to learn sparse attention patterns, Sparsifiner, consists of a normal ViT [15] as the backbone with sparse attention modules at each layer. Our sparse attention module consists of a connectivity mask predictor and a sparse multi-head self-attention (MHSA) module. In both training and inference, we generate a sparse connectivity mask by restricting the number of connections predicted by the mask predictor according to a hyperparameter budget size B. Following this, a sparse MHSA module is used to perform sparse attention based on the connectivity mask. The sparse MHSA module implements an efficient computation using a sparse element-wise product between the full attention map and the sparse connectivity mask to produce a sparse reconstructed attention map. Then, a sparse-dense attention-value product between the sparse reconstructed attention map and the value matrix produces the output of the sparse MHSA module.

For clarity, in the following we describe MHSA for a single attention head only. In practice, we apply the proposed method to each attention head in a ViT. We concatenate the resulting output values from all attention heads and feed them to a linear layer to produce the input to the next transformer layer [37].

ViT Architecture and Naïve MHSA — We base our method on the existing ViT model architecture [15] and naïve implementation of MHSA [37]. A ViT first tokenizes an input image $I \in \mathbb{R}^{h\times w\times 3}$ into a set of n tokens $X \in \mathbb{R}^{n\times d}$, each with dimension d. Each token consists of a patch embedding, retrieved via linear projection of the non-overlapping image patches, and a positional encoding. The resulting sequence of tokens is then fed into MHSA modules to compute the attention matrix $A \in \mathbb{R}^{n\times n}$ as the product of query $Q = X^l W^Q$ and key $K = X^l W^K$ matrices, where $Q, K \in \mathbb{R}^{n\times d}$ and the learned projection matrices $W^Q, W^K \in \mathbb{R}^{d\times d}$ compute query and key as projections of the input $X^l \in \mathbb{R}^{n\times d}$ to layer l. Naïve MHSA then computes the attention matrix A as the softmax of outer product of query and key matrices as shown in the left part of Fig. 2.

Connectivity Mask Predictor — To enable instance-dependent and meaningful attention patterns while limiting the number of connections, we train a connectivity mask predictor and achieve sparsity by thresholding. Specifically, we first compute the low-rank approximation $A^{\text{down}} \in \mathbb{R}^{n\times n_{\text{down}}}$ of the attention matrix A

$$A^{\text{down}} = \text{softmax}\left(\frac{Q(W^{\text{down}}K)^\top}{\sqrt{d}}\right), \tag{1}$$

which we sparsify by thresholding:

$$\tilde{A}^{\text{down}}_{ij} = \begin{cases} A^{\text{down}}_{ij} & \text{if } A^{\text{down}}_{ij} > \tau \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

In the low-rank attention computation (Eq. 1), we first down-project the token dimension of key matrix K to a lower dimension n_{down} using a learned projection matrix $W^{\text{down}} \in \mathbb{R}^{n_{\text{down}}\times n}$. Then, a low-rank approximation of the attention matrix is computed from the outer product of query and down-projected key matrices. Note that in the low-rank attention sparsification (Eq. 2), with a sparse matrix representation we need not explicitly store the zeros.

Next, the connectivity mask predictor (Eq. 3) performs a sparse matrix multiplication of a sparse up-projection matrix $W^{\text{up}} \in \mathbb{R}^{n_{\text{down}}\times n}$ followed by binarization. This produces an up-projected sparse connectivity mask:

$$M = \mathbf{1}\left[\text{Top-}k(\tilde{A}^{\text{down}} W^{\text{up}})\right]. \tag{3}$$

Here, $\tilde{A}^{\text{down}} W^{\text{up}}$ denotes sparse-sparse matrix multiplication, which is efficiently computed. Our key insight is that the post-softmax low-rank attention matrix (Eq. 1) should naturally be sparse. We show an example in Fig. 7.

We apply top-k on the up-projected sparse attention matrix $\tilde{A}^{\text{down}} W^{\text{up}}$, which is the attention connectivity score map. k is set to the budget size B. We discard zero values and binarize to produce a sparse low-rank connectivity mask $M \in \mathbb{R}^{n\times n}$. We indicate binarization by the indicator function $\mathbf{1}[\cdot]$ in the connectivity mask predictor (Eq. 3).

Sparse MHSA — In Fig. 2, we compare our method to naïve MHSA [37] and Linformer [39] in a single head example. In our method, guided by the sparse connectivity mask M, we compute only the nonzero elements of the

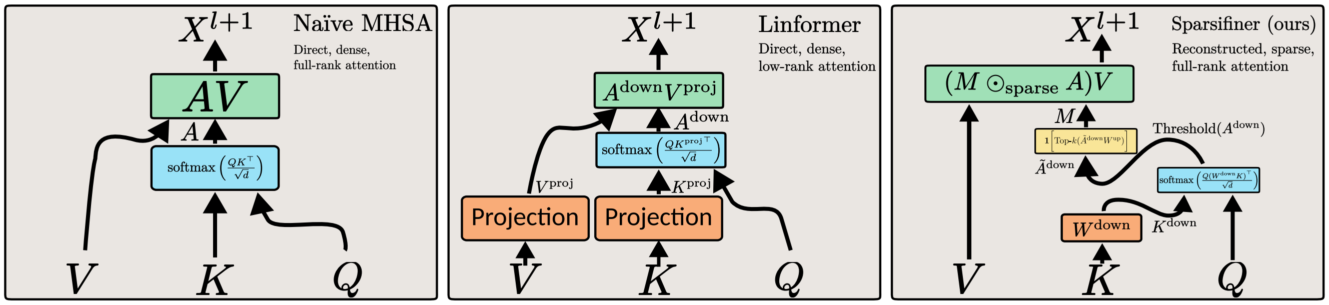

X^{l+1}
Naïve MHSA
Direct, dense, full-rank attention
AV
A
softmax $\left(\frac{QK^\top}{\sqrt{d}}\right)$
V
K
Q
X^{l+1}
Linformer
Direct, dense, low-rank attention
$A^{\text{down}}V^{\text{proj}}$
A^{down}
softmax $\left(\frac{QK^{\text{proj}\top}}{\sqrt{d}}\right)$
V^{proj}
K^{proj}
Projection
Projection
V
K
Q
X^{l+1}
Sparsifiner (ours)
Reconstructed, sparse, full-rank attention
$(M \odot_{\text{sparse}} A)V$
M
$\mathbb{1}\left[\text{Top-}k(\tilde{A}^{\text{down}}W^{\text{up}})\right]$
Threshold(A^{down})
$\tilde{A}^{\text{down}}$
softmax $\left(\frac{Q(W^{\text{down}}K)^\top}{\sqrt{d}}\right)$
W^{down}
K^{down}
V
K
Q

Figure 2. Single head comparison of the MHSA module for naïve MHSA [37], Linformer [39], and Sparsifiner. **Naïve MHSA** incurs quadratic $O(n^2)$ complexity in the number of tokens n. **Linformer** reduces the complexity to linear $O(nn_{\text{down}})$ by using a projection of the key and value matrices to projected key $K^{\text{proj}} \in \mathbb{R}^{n_{\text{down}} \times d}$ and value $V^{\text{proj}} \in \mathbb{R}^{n_{\text{down}} \times d}$ matrices in a low-rank approximation of the attention matrix. **Sparsifiner**'s key insight is to use the low-rank approximation to learn a sparse connectivity mask $M \in \mathbb{R}^{n \times n}$ and sparse up-projection basis W^{up}. Using sparse matrix multiplication, Sparsifiner reduces overall MHSA FLOPs relative to Linformer without restricting the attention matrix to be low rank. Note that in the rightmost column only (Sparsifiner), the attention matrix A is not explicitly constructed, and rather is used to represent sparse attention reconstruction (Eq. 5).

sparse full-rank attention matrix $\tilde{A}$. In order to ensure computational efficiency, we want to have both a sparse up-projection and a sparse low-rank attention matrix. This is equivalent to reconstructing the sparse attention matrix $\tilde{A}$ as an affine combination over a set of sparse basis vectors using a sparse coefficient vector:

$$\tilde{A}_{ij} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)_{ij} \quad \text{iff} \quad M_{ij} = 1. \tag{4}$$

Another way of formulating the sparse full-rank attention matrix is as a sparse element-wise product of the sparse connectivity mask M with the full-rank attention matrix:

$$\tilde{A} = M \odot_{\text{sparse}} A. \tag{5}$$

Here, $\odot_{\text{sparse}}$ is the sparse element-wise product operator, which skips multiplications by zero. Therefore, computing the sparse full-rank attention matrix $\tilde{A}$ (Eq. 4) costs only as many FLOPs as there are nonzero elements in the connectivity mask M. In particular, computing the sparse full-rank attention matrix costs less than the $O(n^2 d)$ required by naïve MHSA.

Finally, Sparsifiner computes a sparse attention-value product using the sparse full-rank attention matrix $\tilde{A}$ and the value matrix V:

$$X^{l+1} = \tilde{A}V. \tag{6}$$

By computing the sparse full-rank attention matrix $\tilde{A}$ (Eq. 4) guided by the sparse connectivity mask, and then computing the sparse attention-value product, we remove the $O(n^2 d)$ complexity required by the naïve MHSA operation. Instead, the sparse MHSA operation in Sparsifiner performs a number of operations proportional to the number of nonzero elements in the connectivity mask M.

Objective functions — The training of Sparsifiner includes training the attention connectivity predictor modules and fine-tuning the backbone to make it adapt to sparse attention. We adopt the standard cross-entropy loss:

$$\mathcal{L}_{\text{cls}} = \text{CrossEntropy}(\mathbf{y}^{\text{pred}}, \mathbf{y}) \tag{7}$$

where $\mathbf{y}^{\text{pred}}$ is the predicted class distribution and $\mathbf{y}$ is the ground-truth class distribution.

To minimize the influence of the attention sparsification process and speed up the convergence of mask predictor, we use another pre-sparsification backbone model as a teacher within a knowledge distillation framework. First, we make the tokens at the last layer close to the ones of the teacher model, where $\mathbf{x}$ and $\mathbf{x}^{\text{teach}}$ are the tokens after the last block of the Sparsifiner and the teacher model, respectively.

$$\mathcal{L}_{\text{distill}}^{\text{token}} = \text{MSE}(\mathbf{x}, \mathbf{x}^{\text{teach}}). \tag{8}$$

Second, we minimize the difference of Sparsifiner and the teacher model's predictions via KL divergence:

$$\mathcal{L}_{\text{distill}}^{\text{cls}} = \text{KL}(\mathbf{y}^{\text{pred}} || \mathbf{y}^{\text{teach}}). \tag{9}$$

Third, we want the connectivity score map generated by the connectivity mask predictor to be a good low-rank approximation of the teacher attention, which can be viewed as knowledge distillation of the attention map. We minimize the Euclidean distance between them:

$$\mathcal{L}_{\text{distill}}^{\text{attn}} = \text{MSE}(\tilde{A}^{\text{down}} W^{\text{up}}, A^{\text{teach}}). \tag{10}$$

Finally, to enforce the sparsity of the up-projection matrix, we use the L_2 regularization. We tried L_1 regularization but found that L_2 gives better training convergence with sufficient sparsity in practice.

$$\mathcal{L}_{\text{spa}} = \sum_i (w_i^{\text{up}})^2 \tag{11}$$

The full training objective combines all objectives:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda^{\text{token}}_{\text{distill}}\mathcal{L}^{\text{token}}_{\text{distill}} + \lambda^{\text{cls}}_{\text{distill}}\mathcal{L}^{\text{cls}}_{\text{distill}} + \lambda^{\text{attn}}_{\text{distill}}\mathcal{L}^{\text{attn}}_{\text{distill}} + \lambda_{\text{spa}}\mathcal{L}_{\text{spa}} \tag{12}$$

Where we set the weight decay as 0.05 in the optimizer instead of directly adding $\lambda_{\text{spa}}\mathcal{L}_{\text{spa}}$ to the objective.

4. Experiments and Results

Implementation details — We train all of the models on the ImageNet dataset [14]. By default, the connectivity mask predictor module is incorporated into every layer of DeiT-S [36] and LV-ViT-S [19]. In all of our experiments, we set the reduced dimension n_{down} to 32 and τ to 0.05 which ensures 87% sparsity ratio of the basis coefficient. The attention budget B is in the range $(0, \text{number of tokens}]$. Budget B is directly determined by the attention keep rate in $(0, 1]$ as the ceiling of the keep rate multiplied by the total number of tokens.

We follow most of the training techniques used in DeiT-S and LV-ViT-S. We use pre-trained ViT models to initialize the backbone models. To improve speed of convergence, we propose a two-phase training strategy. In the first phase, we freeze the backbone model and train the connectivity mask predictor module with attention distillation loss and L2 regularization only. Specifically, we set $\lambda^{\text{token}}_{\text{distill}} = 0.0$, $\lambda^{\text{cls}}_{\text{distill}} = 0.0$, $\lambda^{\text{attn}}_{\text{distill}} = 1.0$ and we also apply a threshold 1e-2 on basis W^{up} to ensure 90% sparsity. We found that this setting helps the connectivity mask predictor to learn W^{up} quickly and loss converges within 5 epochs. In the second phase, we jointly train the backbone model and the connectivity mask predictor module for another 40 epochs. we set $\lambda^{\text{token}}_{\text{distill}} = 0.5$, $\lambda^{\text{cls}}_{\text{distill}} = 0.5$, $\lambda^{\text{attn}}_{\text{distill}} = 0.0$. More details can be found in the supplementary material.

Sparse connectivities and attention visualization — In order to qualitatively investigate the quality of Sparsifiner's sparse attention approximation, we visualize its connectivity mask and sparse reconstructed attention map (Fig. 3). We show the original input image and the connectivity mask of the query patch, where the dark regions represent tokens that are not attended to by the query patch token. For each attention head, Sparsifiner generates a corresponding connectivity mask (Eq. 3). We find that the connectivity mask acts as a region proposal mechanism, which allows different attention heads to locate different informative tokens and gather diverse semantic information. Furthermore, we visualize the sparse attention map efficiently generated using the connectivity mask and compare it with the full attention map. We find that the sparse attention map retains all of the highest connectivity values, while discarding lower connectivity values. Hence the visualizations show that Sparsifiner retains the most salient relations for a given token, while discarding noisy background relations.

Comparison with token pruning — We train and evaluate Sparsifiner on ImageNet and compare to state-of-the-art token pruning baselines (Tab. 1). Since our research question addresses the problem of reducing MHSA complexity, we report trade-offs between top-1 accuracy on ImageNet and computation in terms of MHSA FLOPs. We compare Sparsifiner against baselines by adjusting two hyperparameters: token and attention keep rate. The token keep rate is the fraction of tokens kept in the network at predetermined layers where pruning occurs, which we set according to established token pruning baselines [17, 29]. The attention keep rate is the fraction of attention connectivities at any given MHSA layer, as determined by the connectivity mask predictor (Eq. 3). Hence, varying the attention keep rate reduces FLOPs without necessitating removal of tokens as in token pruning. But both techniques can be combined to achieve complementary effects.

To provide a variety of comparisons we experiment with adding token pruning and Sparsifiner to two common baseline ViT models: DeiT [36] and LV-ViT [19]. On both models, Sparsifiner achieves significant computation savings while maintaining a relatively modest drop in top-1 ac-

Model	Tok. keep rate	Att. keep rate	MHSA (MFLOPs)	Top-1 Acc. (%)
DeiT-S [36]	1.0	1.0	357.7	79.8
EViT [17]	0.7	1.0	193.1 (−46%)	**79.5**
DynamicViT [29]	0.7	1.0	193.1 (−46%)	79.3
Sparsif-EViT (ours)	0.7	0.25	113.3 (−68%)	**79.5**
Sparsifiner (ours)	0.7	0.25	113.3 (−68%)	79.3
EViT [17]	0.5	1.0	149.1 (−58%)	78.5
DynamicViT [29]	0.5	1.0	149.1 (−58%)	77.3
Sparsif-EViT (ours)	0.5	0.25	**86.6** (**−76%**)	78.7
Sparsifiner (ours)	0.5	0.25	**86.6** (**−76%**)	78.4
LV-ViT-S [19]	1.0	1.0	476.9	83.3
EViT-LV-S [17]	0.7	1.0	256.0 (−46%)	83.0
EViT-LV-S [17]	0.5	1.0	198.8 (−58%)	82.5
DynViT-LV-S [29]	0.7	1.0	256.0 (−46%)	83.0
DynViT-LV-S [29]	0.5	1.0	198.8 (−58%)	82.0
Sparsif-LV-S (ours)	1.0	0.5	339.7 (−29%)	**83.4**
Sparsif-LV-S (ours)	1.0	0.25	221.7 (−54%)	83.3
Sparsif-LV-S (ours)	1.0	0.1	**149.5** (**−69%**)	82.8

Table 1. Comparison with token pruning methods on DeiT-S [36] and LV-ViT-S [19] base models. Token pruning methods such as EViT [17] and DynamicViT [29] prune tokens at fixed layers. We show that token pruning methods combine with Sparsifiner's sparse attention connectivities to produce a complementary effect. Sparsifiner combined with EViT [17] achieves a 68% reduction in FLOPs compared with the DeiT-S [36] baseline, while maintaining a top-1 accuracy of 79.5%. Hence Sparsifiner achieves the same top-1 accuracy as EViT [17] with significantly better MHSA FLOPs reduction. The input resolution is 224×224.

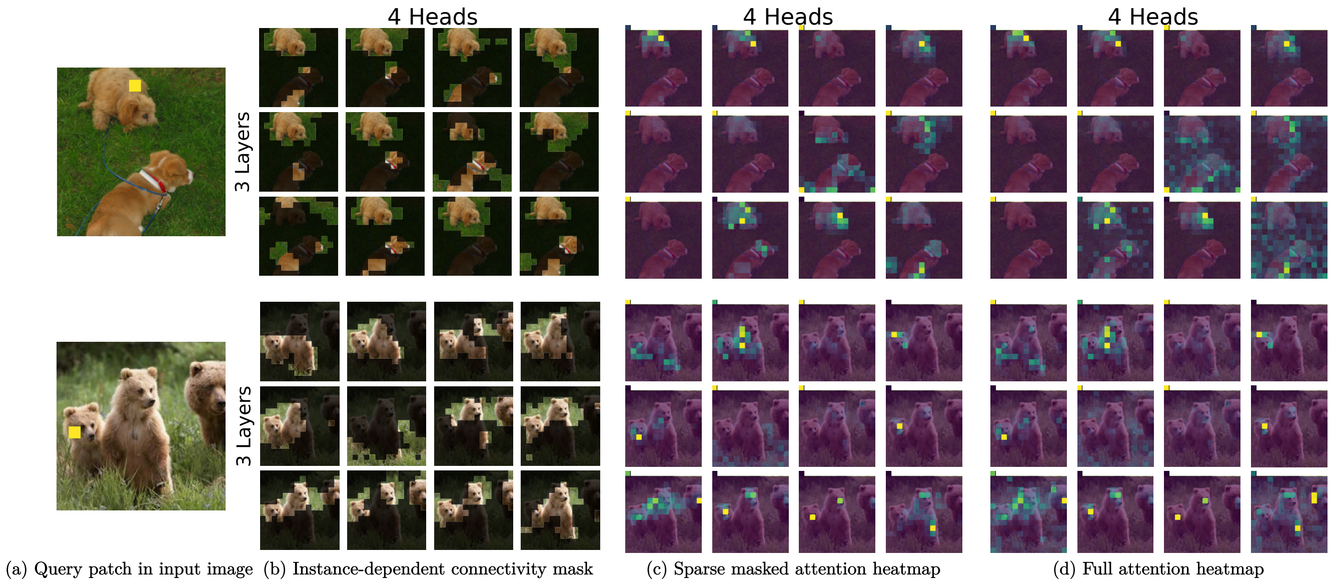

4 Heads
4 Heads
4 Heads
3 Layers
3 Layers
(a) Query patch in input image
(b) Instance-dependent connectivity mask
(c) Sparse masked attention heatmap
(d) Full attention heatmap

Figure 3. Visualization of connectivity mask (b) with sparse (c) and full (d) attention maps for a given query patch (a). In the heatmaps, the blue darker color indicates lower, and yellow brighter color indicates higher attention value. Here we visualize the attention maps for only 3 layers and 4 heads of the ViT. For the dog image (top) we visualize layers 3–5, while for the bear image (bottom) we visualize layers 6–8. We observe that in earlier layers the attention map focuses more on positional information such as nearby tokens, while in later layers semantic relations with distant tokens are more important. For each query patch indicated by a yellow square in the input image, Sparsifiner predicts a sparse connectivity mask using a low-rank approximation to full attention. Using the sparse connectivity mask, Sparsifiner efficiently computes a sparse full-rank attention matrix. By comparison with the rightmost full attention, sparse attention retains all of the most salient relations with the given query patch, while discarding redundant or noisy information in the rest of the image.

curacy. For example, LV-ViT-S [19] trained with Sparsifiner with an attention keep rate of 0.25 reduces the MHSA FLOPs by 53.5% while maintaining the top-1 accuracy of the baseline LV-ViT-S model on ImageNet. When used in combination with token pruning, Sparsifiner achieves an even superior reduction in MHSA FLOPs while maintaining comparable top-1 accuracy to EViT, and superior top-1 accuracy to DynamicViT.

Varying MHSA attention budget — We varied the attention budget of MHSA in order to investigate the tradeoff between MHSA FLOPs and top-1 accuracy for Sparsifiner-S (Tab. 2). The results evaluated on ImageNet show that Sparsifiner-S produces a superior Pareto frontier compared with previous approaches (Fig. 4). In particular, Sparsifiner-S models with attention budgets of 40 and above achieved top-1 accuracy within 0.1% of the full-rank DeiT-S [36] model, while using 58.8% fewer FLOPs in MHSA. Furthermore, Sparsifiner-S models with high attention budgets of 79 and above achieved superior top-1 accuracy compared with the full-rank DeiT-S [36] model, while using fewer FLOPs in MHSA. This suggests that Sparsifiner's sparse full-rank attention reconstruction mechanism induces a useful regularization effect that improves model generalization.

Accelerating ViT on high-resolution images — To show the effectiveness of our method on large input size, we apply our method to DeiT-T [36] with 384 × 384 reso-

Att. keep rate	Att. num.	MHSA (MFLOPs)	Top-1 Acc (%)
1.0 (DeiT-S [36])	197	357.7	79.82
0.9	178	396.8	80.02
0.8	158	360.6	79.97
0.7	138	324.6 (−9%)	79.96
0.6	119	290.3 (−19%)	79.98
0.5	99	254.2 (−29%)	79.94
0.4	79	218.0 (−39%)	79.92
0.3	60	183.6 (−49%)	79.83
0.2	40	147.5 (−59%)	79.71
0.1	20	111.4 (−69%)	79.42
0.05	10	93.3 (−74%)	78.75
0.01	2	78.9 (−78%)	73.03

Table 2. Effect of attention budget on FLOPs and top-1 accuracy. Here the "keep rate" refers to the number of attention connectivities retained at each layer. All other attention connectivities in the sparse full-rank attention matrix (Eq. 4) are set to zero. When keeping only 10 attention connectivities, Sparsifiner produces a top-1 accuracy reduced by only 1.0% compared to the full-attention baseline DeiT-S [36], but with a 73.9% reduction in FLOPs. The input resolution is 224 × 224.

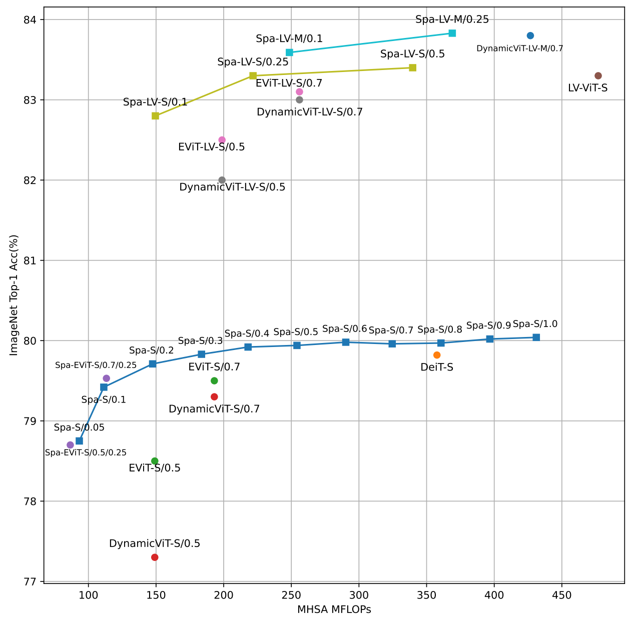


Figure 4. MHSA computation (FLOPs) and top-1 accuracy trade-offs on ImageNet. We compare Sparsifiner (Spa) with the state-of-the-art token pruning methods. Sparsifiner achieves superior trade-offs compared to the baseline. We also report MHSA FLOPs and top-1 accuracy for Sparsifiner-S under varying attention keep rate.

Model	Att. keep rate	MHSA (MFLOPs)	Overall (GFLOPs)	Top-1 Acc (%)
DeiT-T	1.0	1534.1	3.58	75.45
Sparsifiner-T	0.5	851.0 (-45%)	2.89 (-19%)	75.45
Sparsifiner-T	0.25	452.9 (-70%)	2.49 (-30%)	75.35
Sparsifiner-T	0.1	240.5 (-84%)	2.28 (-36%)	74.58

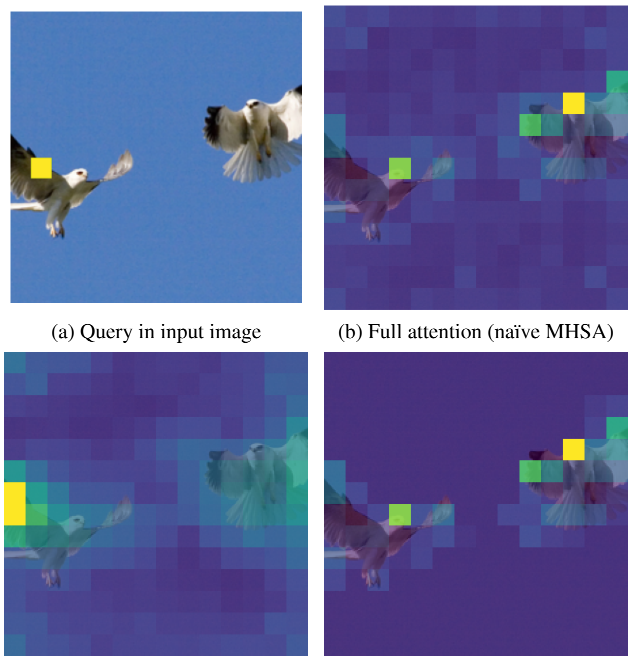
Table 3. Results on high resolution 384×384 images. We apply Sparsifiner on DeiT-T [36] with resolution 384. We show that Sparsifiner reduces the MHSA complexity of DeiT-T-384 [36] by over 84% with modest accuracy drop. Since the number of tokens is quadratic in the resolution, Sparsifiner can reduce a larger portion of MHSA complexity on high-resolution than on 224^2 inputs.

lution (Tab. 3). When dealing with high-resolution images, due to the quadratic complexity in the number of tokens, MHSA becomes expensive compared to feedforward operations. We reduce the MHSA complexity of the DeiT-T [36] model with 384×384 input by over 80% with less than 1% accuracy drop. Our method shows significant potential in accelerating ViT on even higher resolution images where token quantity dominates the model complexity.

Low-rank: connectivities or attention? — Our approach raises the following question: does the utility of the dense low-rank attention matrix come from its use as a connectivity mask? Or is it sufficient to directly use the dense low-rank attention matrix, forgoing the need to reconstruct the sparse full-rank attention matrix (as in the Linformer [39] approach)? We answered this question by comparing the top-1 accuracy of the two approaches (Tab. 4). In this experiment, Sparsifiner-S and Linformer [39] were trained under identical settings, differing only in the attention approximation method. Sparsifiner-S uses a reconstructed sparse full-rank attention matrix, while Linformer uses the dense low-rank attention matrix directly. In order to give both models similar representational capacity, we set the reduced dimension of Linformer [39] equal to the

Model	MHSA (MFLOPs)	Top-1 Acc (%)
Linformer [39]	246.73	77.54
Sparsifiner-S (ours)	224.04	79.79

Table 4. Comparison of sparse full-attention reconstruction with low-rank attention reconstruction. Sparsifiner-S achieves a 2.1% absolute percentage point improvement in top-1 accuracy compared with Linformer [39].



(a) Query in input image (b) Full attention (naïve MHSA)

(c) Low-rank attention (Linformer) (d) Full-rank sparse att. (Sparsifiner)

Figure 5. Low-rank attention (Linformer) and full-rank sparse attention (Sparsifiner) heatmaps. For a given query patch indicated by a yellow square (a), we visualize its low-rank attention map (Linformer) (c) and full-rank sparse attention map (Sparsifiner) (d). Due to discarding the long tail of the attention matrix's eigenspectrum, low-rank attention produces a coarse attention map. By contrast, full-rank sparse attention bears closer resemblance to full attention (b) with low-salience connectivities discarded.

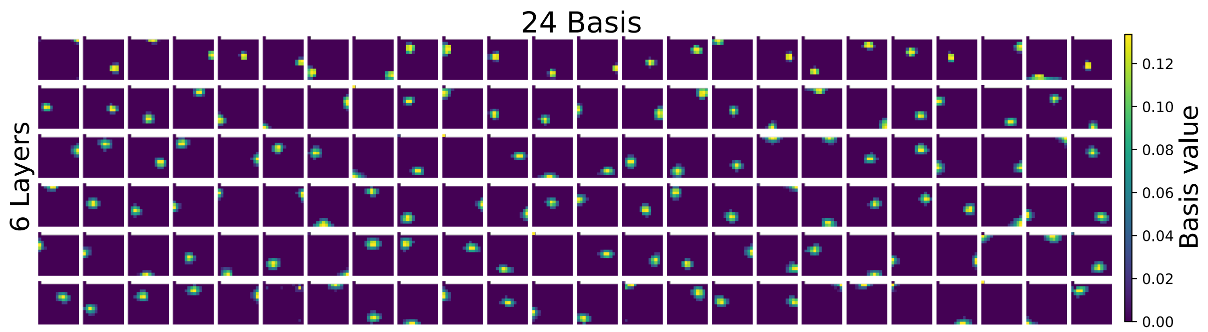


Figure 6. Visualization of the up-projection matrix W^{up} of the first 6 layers of Sparsifiner-S, which we refer to here as a sparse basis. We visualize 24 dimensions of the sparse basis. Dark blue weights indicate low values, which are pruned after training so that only the bright yellow weights are left over. Qualitatively, the sparse basis has a high level of sparsity, making sparse attention reconstruction efficient.

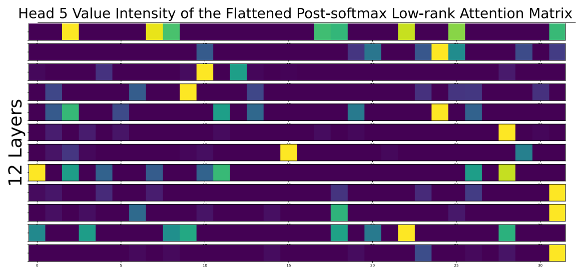


Figure 7. Visualization of the sparse basis coefficient of the 5th attention head over 12 layers of Sparsifiner-S. Dark blue regions indicate low values that are pruned before sparse attention reconstruction during inference, leaving only bright yellow coefficients.

sparse attention budget of Sparsifiner-S. This enforces that the attention-value product of both models' MHSA has the same complexity.

Using the sparse full-rank attention matrix produces a 2.1% absolute percentage point improvement in top-1 accuracy compared with Linformer. This improvement reinforces the superiority of using the low-rank query-key product as a connectivity mask M (Eq. 3), rather than using the low-rank attention matrix A^{down} directly (Eq. 1). Projecting key and value to a lower dimension loses the high-fidelity token-wise information. As a result, the low-rank attention matrix discards the long tail of the full attention matrix's eigenspectrum [39]. In contrast, by using a low-rank connectivity mask to produce a sparse full-rank attention matrix, the long-tail of the full attention matrix's eigenspectrum is preserved. Based on the significant improvement in top-1 accuracy, we conclude that these long-tail eigenvalues are important for model predictive quality in ViTs.

Low- and full-rank attention visualization — In order to further illustrate the qualitative difference between low- and full-rank attention in ViTs, we also present the masked attention heatmap and the full attention heatmap of the query patch (Fig. 5). We show that a connectivity mask can accurately preserve key tokens that are highly related to the query patch and remove the irrelevant ones. As a result, the masked attention heatmap preserves structure and discards noise compared with the full attention heatmap. The visualization results also validate that our Sparsifiner can effectively approximate the full attention ViT.

Sparse low-rank basis and up-projection matrix visualization — To demonstrate that the connectivity mask can be computed by sparse-sparse matrix multiplication, we visualize the up-projection matrix W^{up} of the first six layers of Sparsifiner (Fig. 6). Because the reconstructed sparse attention matrix is a combination of the up-projection matrix's weights, we refer to it as a sparse basis. We show that Sparsifiner naturally learns a sparse basis of local regions resembling 2D Gaussians. For a given token, the sparse bases corresponding to object locations with salient semantic and/or spatial information will activate. Since the sparse attention reconstruction (Eq. 5) is a product of the sparse low-rank attention matrix with the up-projection matrix, we also visualize the post-softmax low-rank attention matrix. Here we view the low-rank attention matrix as a sparse coefficient of the sparse basis (Fig. 7). Qualitatively, the sparse coefficient also exhibits a high degree of sparsity, further validating the efficiency of the sparse attention reconstruction via sparse-sparse matrix multiplication.

5. Conclusions

We presented a computationally efficient approach to learn unstructured, instance-dependent attention in ViTs. The development of sparse attention mechanisms such as the one employed by Sparsifiner opens the door to further research into accelerating sparse ViTs using software-hardware systems approaches. Sparsifiner shows the promise of sparse attention for scaling ViTs to larger and more complex vision tasks. Yet, software-hardware systems approaches are needed to realize its full potential. We hope that our work inspires further research at the intersection of sparse algorithms for ViTs and software-hardware systems approaches to support those sparse algorithms.

References

[1] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020.

[4] 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.

[5] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[6] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. 2022.

[7] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. 2021.

[8] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. In *arXiv:1904.10509*, 2019.

[9] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

[10] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021.

[11] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1601–1610, 2021.

[12] Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pages 4690–4721. PMLR, 2022.

[13] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-scale Hierarchical Image Database. In *CVPR*, 2009.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

[16] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W. Taylor. SSTVOS: Sparse Spatiotemporal Transformers for Video Object Segmentation. In *CVPR*, 2021.

[17] Qihua Feng, Peiya Li, Zhixun Lu, Chaozhuo Li, Zefang Wang, Zhiquan Liu, Chunhui Duan, and Feiran Huang. Evit: Privacy-preserving image retrieval via encrypted vision transformer in cloud computing. *arXiv preprint arXiv:2208.14657*, 2022.

[18] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[19] 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.

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[21] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[22] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[23] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022.

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

[25] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018.

[26] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

[27] Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2555–2565, Online, Nov. 2020. Association for Computational Linguistics.

[28] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

[29] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[30] Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*, 2022.

[31] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[32] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive Space-Time Tokenization for Videos. In *NeurIPS*, 2021.

[33] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *CVPR*, 2022.

[34] 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.

[35] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020.

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training Data-efficient Image Transformers & Distillation through Attention. In *ICML*, 2021.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*. 2017.

[38] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, 2019.

[39] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[40] 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.

[41] 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.

[42] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big Bird: Transformers for Longer Sequences. In *NeurIPS*, 2020.

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

[44] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision*, pages 528–543. Springer, 2020.

[45] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3008, 2021.

[46] Minghang Zheng, Peng Gao, Renrui Zhang, Kunchang Li, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. 2021.

[47] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8739–8748, 2018.

[48] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021.