Chasing Sparsity in Vision Transformers: An End-to-End Exploration

Tianlong Chen[1], Yu Cheng[2], Zhe Gan[2], Lu Yuan[2], Lei Zhang[3], Zhangyang Wang[1]
[1]University of Texas at Austin, [2]Microsoft Corporation, [3]International Digital Economy Academy
{tianlong.chen,atlaswang}@utexas.edu,{yu.cheng,zhe.gan,luyuan}@microsoft.com
leizhangcn@ieee.org

Abstract

Vision transformers (ViTs) have recently received explosive popularity, but their enormous model sizes and training costs remain daunting. Conventional post-training pruning often incurs higher training budgets. In contrast, this paper aims to trim down both the training memory overhead and the inference complexity, without sacrificing the achievable accuracy. We carry out the first-of-its-kind comprehensive exploration, on taking a unified approach of integrating sparsity in ViTs "from end to end". Specifically, instead of training full ViTs, we dynamically extract and train sparse subnetworks, while sticking to a fixed small parameter budget. Our approach jointly optimizes model parameters and explores connectivity throughout training, ending up with one sparse network as the final output. The approach is seamlessly extended from unstructured to structured sparsity, the latter by considering to guide the prune-and-grow of self-attention heads inside ViTs. We further co-explore data and architecture sparsity for additional efficiency gains by plugging in a novel learnable token selector to adaptively determine the currently most vital patches. Extensive results on ImageNet with diverse ViT backbones validate the effectiveness of our proposals which obtain significantly reduced computational cost and almost unimpaired generalization. Perhaps most surprisingly, we find that the proposed sparse (co-)training can sometimes *improve the ViT accuracy* rather than compromising it, making sparsity a tantalizing "free lunch". For example, our sparsified DeiT-Small at (5%, 50%) sparsity for (data, architecture), improves $\mathbf{0.28\%}$ **top-1 accuracy**, and meanwhile enjoys $\mathbf{49.32\%}$ **FLOPs** and $\mathbf{4.40\%}$ **running time** savings. Our codes are available at https://github.com/VITA-Group/SViTE.

1 Introduction

Recent years have seen substantial efforts devoted to scaling deep networks to enormous sizes. Parameter counts are frequently measured in billions rather than millions, with the time and financial outlay necessary to train these models growing in concert. The trend undoubtedly continues with the recent forefront of transformers [1–3] for computer vision tasks. By leveraging self-attention, reducing weight sharing such as convolutions, and feeding massive training data, vision transformers have established many new state-of-the-art (SOTA) records in image classification [1, 2], object detection [4–7], image enhancement [8, 9], and image generation [10–12]. Existing vision transformers and variants, despite the impressive empirical performance, have in general suffered from gigantic parameter-counts, heavy run-time memory usages, and tedious training. That naturally calls for the next step research of slimming their inference and training, without compromising the performance.

Model compression and efficient learning are no strangers to deep learning researchers, although their exploration in the emerging vision transformer field remains scarce [13]. Among the large variety of compression means [14], sparsity has been one of the central themes since the beginning [15].

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Conventional approaches first train dense networks, and then prune a large portion of parameters in the trained networks to zero. Those methods significantly reduce the inference complexity. However, the price is to cost even more significant computational resources and memory footprints at training, since they commonly require (multiple rounds of) re-training to restore the accuracy loss [15–17]. That price becomes particularly prohibitive for vision transformers, whose vanilla one-pass training is already much more tedious, slow, and unstable compared to training standard convolutional networks.

An emerging subfield has explored the prospect of directly training smaller, sparse subnetworks in place of the full networks without sacrificing performance. The key idea is to reuse the sparsity pattern found through pruning and train a sparse network from scratch. The seminal work of lottery ticket hypothesis (LTH) [18] demonstrated that standard dense networks contain sparse matching subnetworks (sometimes called "winning tickets") capable of training in isolation to full accuracy. In other words, we could have trained smaller networks from the start if only we had known which subnetworks to choose. Unfortunately, LTH requires to empirically find these intriguing subnetworks by an iterative pruning procedure [18–27] , which still cannot get rid of the expensiveness of post-training pruning. In view of that, follow-up works reveal that sparsity patterns might emerge at the initialization [28, 29], the early stage of training [30, 31], or in dynamic forms throughout training [32–34] by updating model parameters and architecture typologies simultaneously. These efforts shed light on the appealing prospect of "end to end" efficiency from training to inference, by involving sparsity throughout the full learning lifecycle.

This paper presents the first-of-its-kind comprehensive exploration of integrating sparsity in vision transformers (ViTs) "from end to end". With (dynamic) sparsity as the unified tool, we can improve the inference efficiency from both model and data perspectives, while also saving training memory costs. Our innovative efforts are unfolded along with the following three thrusts:

- **From Dense to (Dynamic) Sparse:** Our primary quest is to find sparse ViTs without sacrificing the achievable accuracy, and meanwhile trimming down the training memory overhead. To meet this challenging demand, we draw inspirations from the latest sparse training works [34, 35] that dynamically extract and train sparse subnetworks instead of training the full models. Sticking to a fixed small parameter budget, our technique jointly optimizes model parameters and explores connectivity throughout the entire training process. We term our first basic approach as *Sparse Vision Transformer Exploration* (**SViTE**).
- **From Unstructured to Structured:** Most sparse training works [32, 33, 36–39, 38, 34, 40, 41, 35] restricted discussion to unstructured sparsity. To attain structured sparsity which is more hardware-friendly, unlike classical channel pruning available for convolutional networks, we customize a first-order importance approximation [16, 42] to guide the prune-and-grow of self-attention heads inside ViTs. This seamlessly extends SViTE to its second variant of *Structured Sparse Vision Transformer Exploration* (**S^{2}ViTE**).
- **From Model to Data:** We further conduct a unified co-exploration towards joint data and architecture sparsity. That is by plugging in a novel learnable token selector to determine the most vital patch embeddings in the current input sample. The resultant framework of *Sparse Vision Transformer Co-Exploration* (**SViTE+**) remains to be end-to-end trainable and can gain additional efficiency.

Extensive experiments are conducted on ImageNet with DeiT-Tiny/Small/Base. Results of substantial computation savings and nearly undamaged accuracies consistently endorse our proposals' effectiveness. Perhaps most impressively, we find that the sparse (co-)training can even *improve the ViT accuracy* rather than compromising it, making sparsity a tantalizing "free lunch". For example, applying SViTE+ on DeiT-Small produces superior compressed ViTs at 50% model sparsity plus 5% data sparsity, saving 49.32% FLOPs and 4.40% running time, while attaining a surprising improvement of 0.28% accuracy; even when the data sparsity increases to 10% (the model sparsity unchanged), there is still no accuracy degradation, meanwhile saving 52.38% FLOPs and 7.63% running time.

2 Related Work

Vision Transformer. Transformer [43] stems from natural language processing (NLP) applications. The Vision Transformer (ViT) [1] pioneered to leverage a pure transformer, to encode an image by splitting it into a sequence of patches, projecting them into token embeddings, and feeding them to

transformer encoders. With sufficient training data, ViT is able to outperform convolution neural networks on various image classification benchmarks [1, 44]. Many ViT variants have been proposed since then. For example, DeiT [2] and T2T-ViT [45] are proposed to enhance ViT's training data efficiency, by leveraging teacher-student and better crafted architectures respectively. In addition to image classification, ViT has attracted wide attention in diverse computer vision tasks, including object detection [4–7], segmentation [46, 47], enhancement [8, 9], image generation [10–12], video understanding [48, 49], vision-language [50–57] and 3D point cloud [58].

Despite the impressive empirical performance, ViTs are generally heavy to train, and the trained models remain massive. That naturally motivates the study to reduce ViT inference and training costs, by considering model compression means. Model compression has been well studied in both computer vision and NLP applications [59–61, 42, 62, 21]. Two concurrent works [13, 63] made initial attempts towards ViT post-training compression by pruning the intermediate features and tokens respectively, but did not jointly consider weight pruning nor efficient training. Another loosely related field is the study of efficient attention mechanisms [64, 10, 52, 65–75]. They mainly reduce the calculation complexity for self-attention modules via various approximations such as low-rank decomposition. Our proposed techniques represent an orthogonal direction and can be potentially combined with them, which we leave as future work. Another latest concurrent work [76] introduced an interpretable module to dynamically and gracefully drop the redundant patches, gaining not only inference efficiency but also interpretability. Being a unique and orthogonal effort from ours, their method did not consider the training efficiency yet.

Pruning and Sparse Training. Pruning is well-known to effectively reduce deep network inference costs [77, 15]. It can be roughly categorized into two groups: (i) unstructured pruning by removing insignificant weight elements per certain criterion, such as weight magnitude [78, 15], gradient [16] and hessian [79]; (ii) structured pruning [80–82] by remove model sub-structures, e.g., channels [80, 81] and attention heads [42], which are often more aligned with hardware efficiency. All above require training the full dense model first, usually for several train-prune-retrain rounds.

The recent surge of sparse training seeks to adaptively identify high-quality sparse subnetworks and train only them. Starting from scratch, those methods learn to optimize the model weights together with sparse connectivity simultaneously. [32, 33] first introduced the Sparse Evolutionary Training (SET) technique [32], reaching superior performance compared to training with fixed sparse connectivity [83, 36]. [37–39] leverages "weight reallocation" to improve performance of obtained sparse subnetworks. Furthermore, gradient information from the backward pass is utilized to guide the update of the dynamic sparse connectivity [38, 34], which produces substantial performance gains. The latest investigations [40, 41, 35] demonstrate that more exhaustive exploration in the connectivity space plays a crucial role in the quality of found sparse subnetworks. Current sparse training methods mostly focus on convolutional networks. Most of them discuss unstructured sparsity, except a handful [84, 30] considering training convolutional networks with structured sparsity.

3 Methodology

Our SViTE method (and its variants S^2ViTE and SViTE+) is inspired from state-of-the-art sparse training approaches [34, 35] in CNNs. This section presents the sparse exploration of ViT architectures, then shows the detailed procedure of input token selection for extra efficiency gains.

3.1 Sparse ViT Exploration

Revisiting sparse training. Sparse training starts from a randomly sparsified model; after optimizing several iterations, it shrinks a portion of parameters based on pre-defined pruning criterion, and activates new connections w.r.t. grow indicators. After upgrading the sparse topology, it trains the new subnetwork until the next update of the connectivity. An illustration of the overall procedure is shown in Figure 1. The key factors of sparse training are ❶ sparsity distribution, ❷ update schedule, ❸ pruning and ❹ grow criterion.

Notations. For a consistent description, we follow the standard notations in [34, 35]. Let $\mathcal{D}$ be the training dataset. $b_t \sim \mathcal{D}$ is a randomly sampled data batch for iteration t. $f_W(\cdot)$ represents the model with parameters $W = (W^{(1)}, \cdots, W^{(L)})$, where $W^{(l)} \in \mathbb{R}^{N_l}, 1 \leq l \leq L$, N_l is the

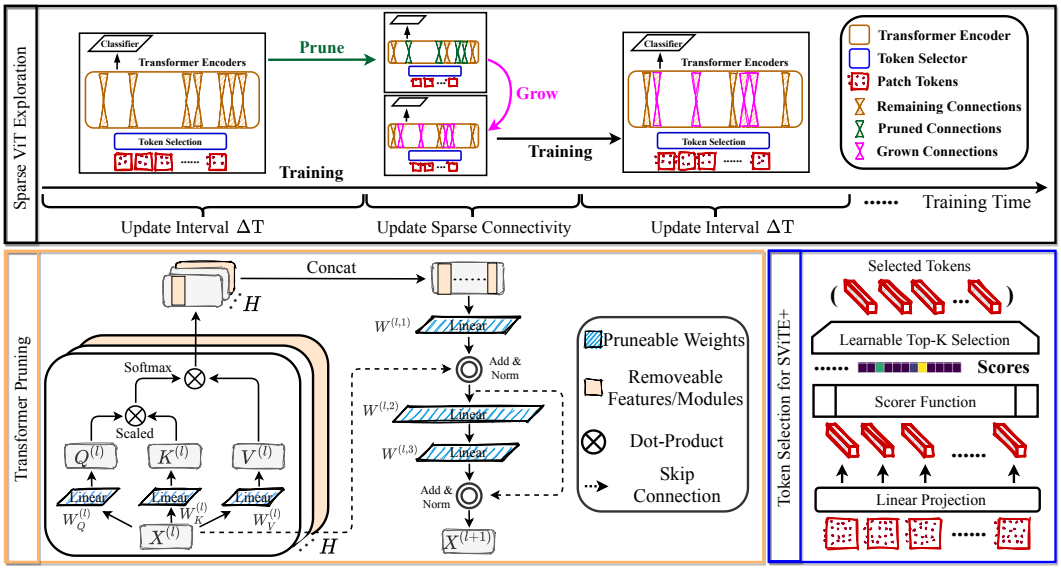

Figure 1: The overall procedure of our proposed sparse ViT exploration framework. ***Upper Figure***: first training ViT for ΔT iterations, then performing prune-and-grow strategies to explore critical sparse connectivities, repreating until convergence. *Bottom Left Figure*: enforcing either structured or unstructured sparsity to transformer layers in ViT. *Bottom Right Figure*: first scoring each input embedding and applying the learnable top-k selection to identify the most informative tokens.

number of prunable parameters in the l_{th} layer, and L denotes the number of transformer layers. Note that the first linear projection layer and the classifier of ViT [1, 2] are not sparsified in our framework. As illustrated in Figure 1(bottom-left), $W_Q^{(l)} = \{W_Q^{(l,h)}\}_{h=1}^{H}$, $W_K^{(l)} = \{W_K^{(l,h)}\}_{h=1}^{H}$, $W_V^{(l)} = \{W_V^{(l,h)}\}_{h=1}^{H}$ are the weights of the self-attention module in the l_{th} layer, $W^{(l,1)}$, $W^{(l,2)}$, $W^{(l,3)}$ are the weights of the multilayer perceptron (MLP) module in the l_{th} layer, and $W^{(l)} = (W_Q^{(l)}, W_K^{(l)}, W_V^{(l)}, W^{(l,1)}, W^{(l,2)}, W^{(l,3)})$ collectively represent all the parameters in the l_{th} layer, where H denotes the number of attention heads, and $1 \leq h \leq H$. $X^{(l)}$, $Q^{(l)}$, $K^{(l)}$, and $V^{(l)}$ are the corresponding input and intermediate features, respectively. Each sparse layer only maintains a fraction $s_l \in (0, 1)$ of its connections, and the overall sparsity of a sparse subnetwork is calculated as the ratio of pruned elements to the total parameter counts, i.e., $\frac{\sum_l s_l \times N_l}{\sum_l N_l}$.

Sparse Vision Transformer Exploration (SViTE). SViTE explores the unstructured sparse topology in vision transformers. To be specific, we adopt *Erdös-Rényi* [32] as our ❶ sparsity distribution. The number of parameters in the sparse layer is scaled by $1 - \frac{n_{l-1}+n_l}{n_{l-1} \times n_l}$, where n_l is the number of neurons at layer l. This distribution allocates higher sparsities to the layers with more parameters by scaling the portion of remaining weights with the sum of the number of output and input neurons/channels. For the ❷ update schedule, it contains: (i) the update interval ΔT, which is the number of training iterations between two sparse topology updates; (ii) the end iteration T_{end}, indicating when to stop updating the sparsity connectivity, and we set T_{end} to 80% of total training iterations in our experiments; (iii) the initial fraction α of connections that can be pruned or grow, which is 50% in our case; (iv) a decay schedule of the fraction of changeable connections $f_{\text{decay}}(t, \alpha, \text{T}_{\text{end}}) = \frac{\alpha}{2}(1 + \cos(\frac{t \times \pi}{\text{T}_{\text{end}}}))$, where a cosine annealing is used, following [34, 35]. During each connectivity update, we choose the weight magnitude as ❸ the pruning indicator, and gradient magnitude as ❹ the grow indicator. Specifically, we eliminate the parameters with the layer-wise smallest weight values by applying a binary mask m_{prune}, then grow new connections with the highest magnitude gradients by generating a new binary mask m_{grow}. Both masks are employed to $W^{(l)}$ via the element-wise dot product, and note that the number of non-zero elements in m_{prune} and m_{grow} are *equal and fixed* across the overall procedure. Newly added connections are not activated in the last sparse topology, and are initialized to zero since it produces better performance as demonstrated in [34, 35].

Infrequent gradient calculation [34] is adopted in our case, which computes the gradients in an online manner and only stores the top gradient values. As illustrated in [34], such fashion amortizes the extra effort of gradient calculation, and makes it still proportional to $1 - s$ as long as $\Delta \mathrm{T} \geq \frac{1}{1-s}$, where s is the overall sparsity.

Structured Sparse Vision Transformer Exploration (S^{2}ViTE). Although models with unstructured sparsity achieve superior performance, *structured* sparsity [80–82] is much more hardware friendly and brings practical efficiency on realistic platforms, which motivates us to propose *Structured Sparse ViT Exploration* (S^{2}ViTE). We inherit the design of ❶ sparsity distribution and ❷ update schedule from the unstructured SViTE, and a round-up function is used to eliminate decimals in the parameter counting. The key differences lie in the new ❸ pruning and ❹ grow strategies.

Pruning criterion: Let $A_{(l,h)}$ denote features computed from the self-attention head $\{W_Q^{(l,h)}, W_K^{(l,h)}, W_V^{(l,h)}\}$ and input embeddings $X^{(l)}$, as shown in Figure 1. We perform the Taylor expansion to the loss function [16, 42], and derive a proxy score for head importance blow:

$$\mathcal{I}_p^{(l,h)} = \left| A_{(l,h)}^{\mathrm{T}} \cdot \frac{\partial \mathcal{L}(X^{(l)})}{\partial A_{(l,h)}} \right|, \quad (1)$$

where $\mathcal{L}(\cdot)$ is the cross-entropy loss as used in ViT. During each topology update, we remove attention heads with the smallest $\mathcal{I}_p^{(l,h)}$. For MLPs, we score neurons with ℓ_1-norm of their associated weight vectors [85], and drop insignificant neurons. For example, the j_{th} neuron of $W^{(l,1)}$ in Figure 1 has an importance score $\|W_{j,\cdot}^{(l,1)}\|_{\ell_1}$, where $W_{j,\cdot}^{(l,1)}$ is the j_{th} row of $W^{(l,1)}$.

Grow criterion: Similar to [34, 35], we active the new units with the highest magnitude gradients, such as $\|\frac{\partial \mathcal{L}(X^{(l)})}{\partial A_{(l,h)}}\|_{\ell_1}$ and $\|\frac{\partial \mathcal{L}(X^{(l)})}{\partial W_{j,\cdot}^{(l,1)}}\|_{\ell_1}$ for the h_{th} attention head and the j_{th} neuron of the MLP ($W^{(l,1)}$), respectively. The gradients are calculated in the same manner as the one in unstructured SViTE, and newly added units are also initialized to zero.

Algorithm 1 Sparse ViT Co-Exploration (SViTE+).

Initialize: ViT model f_W, Dataset $\mathcal{D}$, Sparsity distribution $\mathbb{S} = \{s_1, \cdots, s_L\}$, Update schedule $\{\Delta \mathrm{T}, \mathrm{T}_{\mathrm{end}}, \alpha, f_{\mathrm{decay}}\}$, Learning rate η

Initialize f_W with random sparsity $\mathbb{S}$ ▷ *Highly reduced parameter count.*
for each training iteration t **do**
  Sampling a batch $b_t \sim \mathcal{D}$
  Scoring the input token embeddings and selecting the top-k informative tokens ▷ *Token selection*
  if (t mod $\Delta \mathrm{T} == 0$) and $t < \mathrm{T}_{\mathrm{end}}$ **then**
    for each layer l **do**
      $\rho = f_{\mathrm{decay}}(t, \alpha, \mathrm{T}_{\mathrm{end}}) \cdot (1 - s_l) \cdot N_l$
      Performing prune-and-grow with portion ρ w.r.t. certain criterion, generating masks m_{prune} and m_{grow} to update f_W's sparsity patterns ▷ *Connectivity exploration*
    end for
  else
    $W = W - \eta \cdot \nabla_W \mathcal{L}_t$ ▷ *Updating Weights*
  end if
end for
return a sparse ViT with a trained token selector

3.2 Data and Architecture Sparsity Co-Exploration for Higher Efficiency

Besides exploring sparse transformer architectures, we further slim the dimension of input token embeddings for extra efficiency bonus by leveraging a learnable token selector, as presented in Figure 1. Meanwhile, the introduced data sparsity also serves as an implicit regularization for ViT training, which potentially leads to improved generalization ability, as evidenced in Table 6. Note that, due to skip connections, the number of input tokens actually determines the dimension of intermediate features, which substantially contributes to the overall computation

Algorithm 2 The top-k selector in a PyTorch-like style.

```
def topk_selector(logits, k, tau, dim=-1):
# Maintain tokens with the top-$k$ highest scores
    gumbels =
        -torch.empty_like(logits).exponential_().log()
    gumbels = (logits + gumbels) / tau
    # tau is the temperature
    y_soft = gumbels.softmax(dim)
    # Straight through
    index = y_soft.topk(k, dim=dim)[1]
    y_hard = scatter(logits, index, k)
    ret = y_hard - y_soft.detach() + y_soft
    return ret
```

Table 1: Details of training configurations in our experiments, mainly following the settings in [2].

Backbone	Update Schedule $\{\Delta\mathrm{T}, \mathrm{T}_{\mathrm{end}}, \alpha, f_{\mathrm{decay}}\}$	Batch Size	Epochs	Inherited Settings from DeiT [2]
DeiT-Tiny	$\{20000, 1200000, 0.5, \mathrm{cosine}\}$	512	600	AdamW, $0.0005 \times \frac{\mathrm{batchsize}}{512}$, cosine decay
DeiT-Small	$\{15000, 1200000, 0.5, \mathrm{cosine}\}$	512	600	warmup 5 epochs, 0.05 weight decay
DeiT-Base	$\{7000, 600000, 0.5, \mathrm{cosine}\}$	1024	600	0.1 label smoothing, augmentations, etc.

cost. In other words, the slimmed input token embeddings directly result in compressed intermediate features, and bring substantial efficiency gains.

For the input tokens $X^{(1)} \in \mathbb{R}^{n \times d}$, where n denotes the number of tokens to be shrunk, and d is the dimension of each token embedding that keeps unchanged. As shown in Figure 1, all token embeddings are passed through a learnable scorer function which is parameterized by an MLP in our experiments. Then, a selection of the top-k importance scores ($1 \leq k \leq d$) is applied on top of it, aiming to preserve the significant tokens and remove the useless ones. To optimize parameters of the scorer function, we introduce the popular Gumbel-Softmax [86, 87] and straight-through tricks [88] to enable gradient back-propagation through the top-k selection, which provides an efficient solution to draw samples from a discrete probability distribution. A detailed implementation is in Algorithm 2.

The full pipeline of data and architecture co-exploration is summarized in Algorithm 1. We term this approach SViTE+. We first feed the randomly sampled data batch to the token selector and pick the top-k informative token embeddings. Then, we alternatively train the sparse ViT for $\Delta\mathrm{T}$ iterations and perform prune-and-grow to explore the sparse connectivity in ViTs dynamically. In the end, a sparse ViT model with a trained token selector is returned and ready for evaluation.

4 Experiments

Baseline pruning methods. We extend several effective pruning methods from CNN compression as our strong baselines. *Unstructured pruning*: (i) One-shot weight Magnitude Pruning (OMP) [15], which removes insignificant parameters with the globally smallest weight values; (ii) Gradually Magnitude Pruning (GMP) [17], which seamlessly incorporates gradual pruning techniques within the training process by eliminating a few small magnitude weights per iteration; and (iii) Taylor Pruning (TP) [16], which utilizes the first-order approximation of the training loss to estimate units' importance for model sparsification. *Structured pruning:* Salience-based Structured Pruning (SSP). We draw inspiration from [42, 85], and remove sub-modules in ViT (e.g., self-attention heads) by leveraging their weight, activation, and gradient information. Moreover, due to the repetitive architecture of ViT, we can easily reduce the number of transformer layers to create a smaller dense ViT (Small-Dense) baseline that has similar parameter counts to the pruned ViT model.

Implementation details. Our experiments are conducted on ImageNet with DeiT-Tiny/Small/Base backbones. The detailed training configurations are listed in Table 1, which mainly follows the default setups in [2]. All involved customized hyperparameters are tuned via grid search (later shown in Figure 3). For a better exploration of sparsity connectivities, we increase training epochs to 600 for all experiments. GMP [17] has an additional hyperparameter, i.e., the pruning schedule, which starts from $\frac{1}{6}$ and ends at $\frac{1}{2}$ of the training epochs with 20 times pruning in total. More details are referred to Appendix A1.

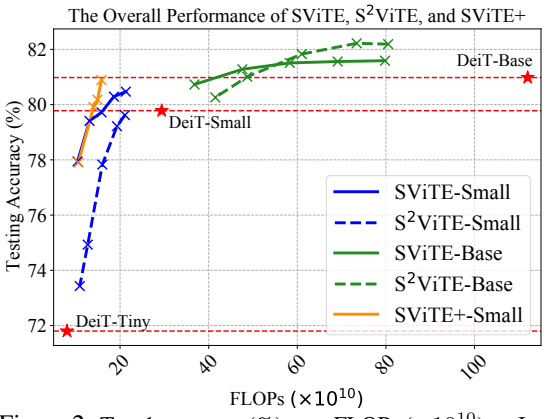

Figure 2: Top-1 accuracy (%) over FLOPs ($\times 10^{10}$) on ImageNet of our methods, i.e., SViTE, S^{2}ViTE, and SViTE+, compared to DeiT baselines, trained on Imagenet-1K only.

Training time measuring protocol. We strictly measure the running time saving of (sparse) vision transformers on the ImageNet-1K task using CUDA benchmark mode. To be specific, we separately calculate the time elapsed during each iteration, to eliminate the impact of the hardware environment as much as possible. Note that the time for the data I/O is excluded.

Highlight of our findings. The overall performance of SViTE, S^{2}ViTE, and SViTE+ on DeiT backbones are summarized in Figure 2. We highlight some takeaways below.

Takeaways: ❶ SViTE produces sparse DeiTs with enhanced generalization and substantial reduced FLOPs, compared to its dense counterpart (⋆). SViTE+ further improves the performance of SViTE by selecting the most vital patches. ❷ S^{2}ViTE achieves matched accuracy on DeiT-Small, and significantly enhances performance on DeiT-Base. Meanwhile, its structural sparsity brings considerable running time savings. ❸ Appropriate data and architecture sparsities can effectively regularize ViT training, leading to a new SOTA win-win between ViT accuracy and efficiency.

4.1 SViTE with Unstructured Sparsity

We perform SViTE to mine vital unstructured sparsity in DeiTs [2]. Solid lines in Figure 2 record the top-1 test-set accuracy over FLOPs on ImageNet-1K of SViTE-Small and SViTE-Base with a range of sparsity from 30% to 70%. In general, we observe that SViTE generates superior sparse ViTs with both accuracy and efficiency gains. Table 2, 3, and 5 present the comparison between SViTE and various pruning baselines. From these extensive results, we draw several consistent observations. First, compared to the dense baselines, SViTE-Tiny, -Small, and -Base obtain $25.56\% \sim 34.16\%$, $46.26\% \sim 55.44\%$, and $47.95\% \sim 57.50\%$ FLOPs reduction, respectively, at $30\% \sim 60\%$ sparsity levels with only a negligible accuracy drop within 0.5%. It verifies the effectiveness of our proposal, and indicates severe parameter redundancy in ViT. Second, our SViTE models from dynamic explorations consistently surpass other competitive baseline methods, including OMP, GMP, TP, and Small-Dense by a substantial performance margin. Among all the baseline approaches, GMP that advocates a gradual pruning schedule achieves the best accuracy with all three DeiT backbones. Third, in Figure 2, both SViTE-Small (blue solid line) and SViTE-Base (green solid line) show an improved trade-off between accuracy and efficiency, compared to their dense DeiT counterparts. Interestingly, we also observe that with similar parameter counts, a large sparse ViT consistently outperforms the corresponding smaller dense ViT. A possible explanation is those appropriate sparse typologies regularize network training and lead to enhanced generalization, which coincides with recent findings of critical subnetworks (i.e., winning tickets) in dense CNNs [89, 22] and NLP transformer [21, 90] models.

Table 2: Results of **SViTE-Tiny** on ImageNet-1K. Accuracies (%) within/out of parenthesis are the reproduced/reported [2] performance.

Models	Sparsity (#Para.)	FLOPs Saving	Accuracy (%)
DeiT-Tiny	0% (5.72M)	0%	72.20 (71.80)
SViTE-Tiny	30% (4.02M)	25.56%	71.78
OMP	30% (4.02M)	25.56%	68.35
GMP	30% (4.02M)	25.56%	69.56
TP	30% (4.02M)	25.56%	68.38
SViTE-Tiny	40% (3.46M)	34.16%	71.75
OMP	40% (3.46M)	34.16%	66.52
GMP	40% (3.46M)	34.15%	68.36
TP	40% (3.46M)	34.17%	65.45
Small-Dense	0% (3.94M)	32.54%	67.33

Table 3: Results of **SViTE-Small** on ImageNet-1K. Accuracies (%) within/out of parenthesis are the reproduced/reported [2] performance.

Models	Sparsity (#Para.)	FLOPs Saving	Accuracy (%)
DeiT-Small	0% (22.1M)	0%	79.90 (79.78)
SViTE-Small	50% (11.1M)	46.26%	79.72
OMP	50% (11.1M)	46.25%	76.32
GMP	50% (11.1M)	46.26%	76.88
TP	50% (11.1M)	46.26%	76.30
SViTE-Small	60% (8.9M)	55.44%	79.41
OMP	60% (8.9M)	55.44%	75.32
GMP	60% (8.9M)	55.44%	76.79
TP	60% (8.9M)	55.44%	74.50
Small-Dense	0% (11.4M)	49.32%	73.93

Table 4: Results of S^{2}ViTE with structured sparsity on ImageNet-1K with DeiT-Tiny/Small/Base. Accuracies (%) within/out of parenthesis are the reproduced/reported [2] performance.

Models	Sparsity (%)	Parameters	FLOPs Saving	Running Time Reduced	Top-1 Accuracy (%)
DeiT-Tiny (Dense)	0%	5.72M	0%	0%	72.20 (71.80)
SViTE-Tiny (Unstructured)	30%	4.02M	25.56%	0%	71.78
SSP-Tiny (Structured)	30%	4.21M	23.69%	10.57%	68.59
S^{2}ViTE-Tiny (Structured)	30%	4.21M	23.69%	10.57%	70.12
DeiT-Small (Dense)	0%	22.1M	0%	0%	79.90 (79.78)
SViTE-Small (Unstructured)	40%	13.3M	36.73%	0%	80.26
SSP-Small (Structured)	40%	14.6M	31.63%	22.65%	77.74
S^{2}ViTE-Small (Structured)	40%	14.6M	31.63%	22.65%	79.22
DeiT-Base (Dense)	0%	86.6M	0%	0%	81.80 (80.98)
SViTE-Base (Unstructured)	40%	52.0M	38.30%	0%	81.56
SSP-Base (Structured)	40%	56.8M	33.13%	24.70%	80.08
S^{2}ViTE-Base (Structured)	40%	56.8M	33.13%	24.70%	82.22

4.2 S^{2}ViTE with Structured Sparsity

For more practical benefits, we investigate sparse DeiTs with structured sparsity. Results are summarized in Table 4. Besides the obtained $23.79\% \sim 33.63\%$ FLOPs savings, S^{2}ViTE-Tiny, S^{2}ViTE-

Small, and S^{2}ViTE-Base enjoy an extra 10.57%, 22.65%, and 24.70% running time reduction, respectively, from $30\% \sim 40\%$ structured sparsity with competitive top-1 accuracies. Furthermore, S^{2}ViTE consistently outperforms the baseline structured pruning method (SSP), which again demonstrates the superior sparse connectivity learned from dynamic sparse training.

The most impressive results come from S^{2}ViTE-Base at 40% structured sparsity. It even surpasses the dense DeiT base model by $0.42\% \sim 1.24\%$ accuracy with 34.41% parameter counts, 33.13% FLOPs, and 24.70% running time reductions. We conclude that (i) an adequate sparsity from S^{2}ViTE boosts ViT's generalization ability, which can be regarded as an implicit regularization; (ii) larger ViTs (e.g., DeiT-Base) tend to have more superfluous self-attention heads, and are more amenable to structural sparsification from S^{2}ViTE, based on Figure 2 where dash lines denote the overall performance of S^{2}ViTE-Small and S^{2}ViTE-Base with a range of sparsity from 30% to 70%.

Table 5: Results of **SViTE-Base** on ImageNet-1K. Accuracies (%) within/out of parenthesis are the reproduced/reported [2] performance.

Models	Sparsity (#Para.)	FLOPs Saving	Accuracy (%)
DeiT-Base	0% (86.6M)	0%	81.80 (80.98)
SViTE-Base	50% (43.4M)	47.95%	81.51
OMP	50% (43.4M)	47.94%	80.26
GMP	50% (43.4M)	47.95%	80.79
TP	50% (43.4M)	47.94%	80.55
SViTE-Base	60% (34.8M)	57.50%	81.28
OMP	60% (34.8M)	57.50%	80.25
GMP	60% (34.8M)	57.50%	80.44
TP	60% (34.8M)	57.49%	80.37
Small-Dense	0% (44.0M)	49.46%	78.59

Table 6: Results of **SViTE+-Small** on ImageNet-1K. Accuracies (%) within/out of parenthesis are the reproduced/reported [2] performance.

#Tokens (%)	Time Reduced	FLOPs Saving	Accuracy (%)
SViTE+-Small 50% Unstructured Sparsity			
100%	0%	46.26%	79.72
95%	4.40%	49.32%	80.18
90%	7.63%	52.38%	79.91
70%	19.77%	63.95%	77.90
S^{2}ViTE+-Small 40% Structured Sparsity			
100%	22.65%	31.63%	79.22
95%	27.17%	37.76%	78.44
90%	29.21%	41.50%	78.16
70%	39.10%	54.96%	74.77

4.3 SViTE+ with Data and Architecture Sparsity Co-Exploration

In this section, we study data and architecture sparsity co-exploration for ViTs, i.e., SViTE+. Blessed by the reduced input token embeddings, even ViTs with unstructured sparsity can have running time savings. The benefits are mainly from the shrunk input and intermediate feature dimensions. Without loss of generality, we consider SViTE+-Small with 50% unstructured sparsity and S^{2}ViTE+-Small with 40% structured sparsity as examples. As shown in Table 6 and Figure 2, SViTE+-Small at 50% unstructured sparsity is capable of abandoning $5\% \sim 10\%$ tokens while achieving $4.40\% \sim 7.63\%$ running time and $49.32\% \sim 52.38\%$ FLOPs savings, with even improved top-1 testing accuracy. It again demonstrates that data sparsity as an implicit regularizer plays a beneficial role in ViT training. However, slimming input and intermediate embedding is less effective when incorporated with S^{2}ViTE, suggesting that aggressively removing structural sub-modules hurts ViT's generalization.

4.4 Ablation and Generalization Study of SViTEs

Update interval in SViTE The length of the update interval ΔT controls one of the essential trade-

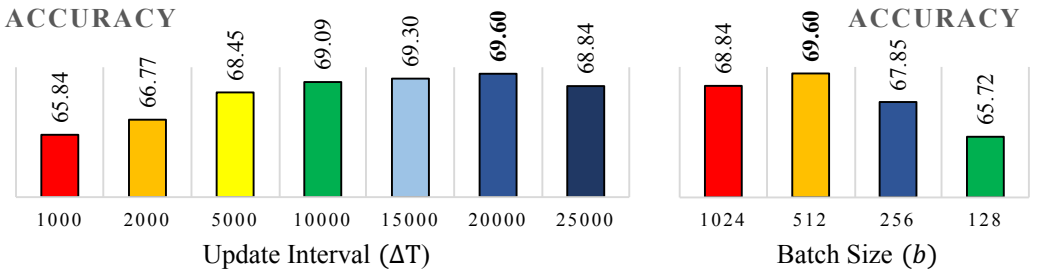

Figure 3: Accuracy of SViTE-Tiny with 50% unstructured sparsity. *Left*: ablation studies of the update interval (ΔT); *Right*: ablations studies of the adopted batch size (b).

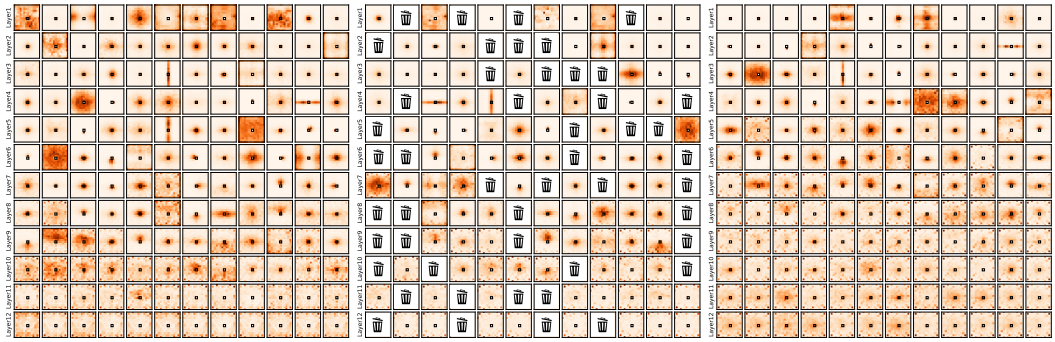

Figure 4: Attention probabilities for DeiT-Base, S^{2}ViTE-Base, and SViTE-Base models with 12 layers (rows) and 12 heads (columns) using visualization tools provided in [94]. Attention maps are averaged over 100 test samples from ImageNet-1K to present head behavior and remove the dependence on the input content. The black square is the query pixel. 🗑 indicates pruned attention heads. Zoom-in for better visibility.

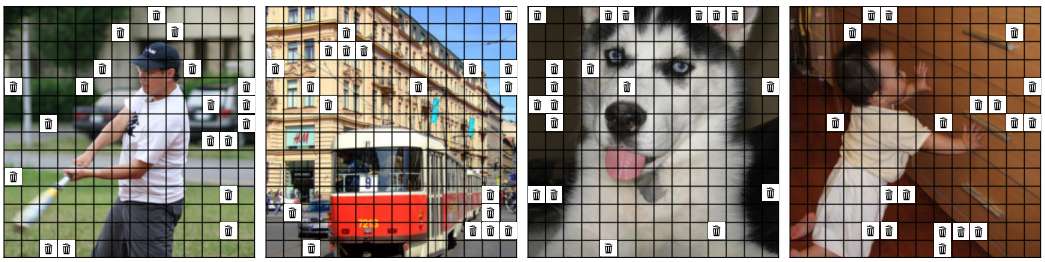

Figure 5: Learned patch selection patterns of SViTE+-Small at 10% data and 50% architecture sparsity levels. 🗑 indicates removed inessential patches.

Batch size in SViTE. Besides the update interval ΔT, batch size (b) also affects the aforementioned trade-off, especially for the data-hungry ViT training. We investigate different batch sizes in Figure 3 (*Right*), and find that $b = 512$ outperforms other common options for SViTE-Tiny.

Generalization study of SViTE and its variants. It is worth mentioning that our proposed frameworks (SViTE, S^{2}ViTE, SViTE+) are independent of the backbone architectures, and can be easily plugged in other vision transformer models [91, 45, 92, 93]. We implemented both SViTE and S^{2}ViTE on TNT-S [91]. SViTE-TNT-S gains 0.13 accuracy improvements (Ours: 81.63 v.s. TNT-S: 81.50) and 37.54% FLOPs savings at 40% unstructured sparsity; S^{2}ViTE-TNT-S obtains 32.96% FLOPs and 23.71% running time reductions at 40% structured sparsity with almost unimpaired accuracy (Ours: 81.34 v.s. TNT-S:81.50).

4.5 Visualization

Sparse connectivity patterns. We provide unit-wise and element-wise heatmap visualizations for SViTE-Base with 40% structured sparsity in Figure A7 (in Appendix). Similarly, element-wise heatmap visualizations of SViTE-Base with 50% unstructured sparsity are displayed in Figure A6. We find that even unstructured sparsity exploration can develop obvious structural patterns (i.e., "vertical lines" in mask heatmaps), which implies a stronger potential for hardware speedup [95].

Self-attention heatmaps. As shown in Figure 4, we utilize tools in [94] to visualize attention maps of (sparse) ViTs. Multiple attention heads show similar behaviors, which implies the structural redundancy. Fortunately, S^{2}ViTE eliminates unnecessary heads to some extent. With regard to SViTE-Base's visual results, it seems to activate fewer attention heads for predictions (darker colors mean larger values), compared to the ones of dense DeiT-Base. We also observe that in the bottom layers, the attention probabilities are more centered at several heads; while in the top layers, the attention probabilities are more uniformly distributed. This kind of tendency is well preserved by our sparse ViT (SViTE) from Dense ViTs.

Learned patch selection patterns. Figure 5 presents the learned behaviors of our token selector in SViTE+. We observe that the useless removed patches are typically distributed around the main object

or in the background. Meanwhile, the patches within the objects of interest are largely persevered, which evidences the effectiveness of our learned patch token selector.

5 Conclusion and Discussion of Broader Impact

In this work, we introduce sparse ViT exploration algorithms, SViTE, and its variants S^2ViTE and SViTE+, to explore high-quality sparse patterns in both ViT's architecture and input token embeddings, alleviating training memory bottleneck and pursuing inference ultra-efficiency (e.g., running time and FLOPs). Comprehensive experiments on ImageNet validate the effectiveness of our proposal. Our informative visualizations further demonstrate that SViTE+ is capable of mining crucial connections and input tokens by eliminating redundant units and dropping useless token embeddings. Future work includes examining the performance of our sparse ViTs on incoming hardware accelerators [96–100], which will provide better supports for sparsity.

This work is scientific in nature, and we do not believe it has immediate negative societal impacts. Our findings of sparse vision transformers are highly likely to reduce both memory and energy costs substantially, leading to economic deployment in real-world applications (e.g., on smartphones).

Acknowledgment

Z.W. is in part supported by an NSF RTML project (#2053279).

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[3] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020.

[4] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020.

[5] 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.

[6] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. *arXiv preprint arXiv:2011.09094*, 2020.

[7] 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.

[8] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

[9] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020.

[10] 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.

[11] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020.

[12] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 2021.

[13] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning, 2021.

[14] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

[16] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019.

[17] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. *arXiv preprint arXiv:1912.05671*, 2019.

[20] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang. Efficient lottery ticket finding: Less data is more. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12380–12390. PMLR, 18–24 Jul 2021.

[21] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*, 2020.

[22] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. *arXiv preprint arXiv:2012.06908*, 2020.

[23] Xuxi Chen, Zhenyu Zhang, Yongduo Sui, and Tianlong Chen. {GAN}s can play lottery tickets too. In *International Conference on Learning Representations*, 2021.

[24] Haoyu Ma, Tianlong Chen, Ting-Kuei Hu, Chenyu You, Xiaohui Xie, and Zhangyang Wang. Good students play big lottery better. *arXiv preprint arXiv:2101.03255*, 2021.

[25] Zhe Gan, Yen-Chun Chen, Linjie Li, Tianlong Chen, Yu Cheng, Shuohang Wang, and Jingjing Liu. Playing lottery tickets with vision and language. *arXiv preprint arXiv:2104.11832*, 2021.

[26] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. *arXiv preprint arXiv:2102.06790*, 2021.

[27] Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Ultra-data-efficient gan training: Drawing a lottery ticket first, then training it toughly. *arXiv preprint arXiv:2103.00397*, 2021.

[28] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

[29] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

[30] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Toward more efficient training of deep networks. In *International Conference on Learning Representations*, 2020.

[31] Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*, 2020.

[32] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[33] Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 2020.

[34] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[35] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. *arXiv preprint arXiv:2102.02887*, 2021.

[36] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.

[37] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, 2019.

[38] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[39] Shiwei Liu, Decebal Constantin Mocanu, Yulong Pei, and Mykola Pechenizkiy. Selfish sparse rnn training. *arXiv preprint arXiv:2101.09048*, 2021.

[40] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33, 2020.

[41] Md Aamir Raihan and Tor M Aamodt. Sparse weight activation training. *arXiv preprint arXiv:2001.01969*, 2020.

[42] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one?, 2019.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[44] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

[45] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[46] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020.

[47] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020.

[48] 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.

[49] 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.

[50] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.

[51] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[52] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020.

[53] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[54] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[55] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11336–11344, 2020.

[56] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[57] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13041–13049, 2020.

[58] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020.

[59] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

[60] Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*, 2019.

[61] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. Compressing large-scale transformer-based models: A case study on bert. *arXiv preprint arXiv:2002.11985*, 2020.

[62] J. S. McCarley, Rishav Chakravarti, and Avirup Sil. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360*, 2019.

[63] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. *arXiv preprint arXiv:2106.02852*, 2021.

[64] 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. *arXiv preprint arXiv:2103.15358*, 2021.

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

[66] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[67] 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.

[68] 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.

[69] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

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

[71] 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.

[72] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[73] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[74] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

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

[76] Bowen Pan, Yifan Jiang, Rameswar Panda, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers, 2021.

[77] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[78] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015.

[79] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann, 1990.

[80] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[81] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[82] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[83] Decebal Constantin Mocanu, Elena Mocanu, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. A topological insight into restricted boltzmann machines. *Machine Learning*, 104(2-3):243–270, 2016.

[84] Sangkug Lym, Esha Choukse, Siavash Zangeneh, Wei Wen, Sujay Sanghavi, and Mattan Erez. Prunetrain: fast neural network training by dynamic sparse model reconfiguration. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2019.

[85] Brian R Bartoldson, Ari S Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *arXiv preprint arXiv:1906.03728*, 2019.

[86] Emit J. Gumbel. Statistical theory of extreme values and some practical applications. *The Journal of the Royal Aeronautical Society*, 58(527):792–793, 1954.

[87] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. *arXiv preprint arXiv:1411.0030*, 2014.

[88] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

[89] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. The lottery ticket hypothesis at scale. *arXiv preprint arXiv:1903.01611*, 2019.

[90] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[91] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[92] 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. *arXiv preprint arXiv:2102.12122*, 2021.

[93] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

[94] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

[95] Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14629–14638, 2020.

[96] Peiqi Wang, Yu Ji, Chi Hong, Yongqiang Lyu, Dongsheng Wang, and Yuan Xie. Snrram: An efficient sparse neural network computation architecture based on resistive random-access memory. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, 2018.

[97] Mike Ashby, Christiaan Baaij, Peter Baldwin, Martijn Bastiaan, Oliver Bunting, Aiken Cairncross, Christopher Chalmers, Liz Corrigan, Sam Davis, Nathan van Doorn, et al. Exploiting unstructured sparsity on next-generation datacenter hardware. *None*, 2019.

[98] Chen Liu, Guillaume Bellec, Bernhard Vogginger, David Kappel, Johannes Partzsch, Felix Neumärker, Sebastian Höppner, Wolfgang Maass, Steve B Furber, Robert Legenstein, et al. Memory-efficient deep learning on a spinnaker 2 prototype. *Frontiers in neuroscience*, 12:840, 2018.

[99] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254, 2016.

[100] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.