LPViT: Low-Power Semi-structured Pruning for Vision Transformers

Kaixin Xu[1,2*], Zhe Wang[1,2*], Chunyun Chen[2], Xue Geng[1], Jie Lin[1], Xulei Yang[1], Min Wu[1] ✉, Xiaoli Li[1], and Weisi Lin[2]

[1] Institute for Infocomm Research (I^{2}R), Agency for Science, Technology and Research (A*STAR), 1 Fusionopolis Way, 138632, Singapore
{xuk,wang_zhe,geng_xue,yang_xulei,wumin,xlli}@i2r.a-star.edu.sg
jie.dellinger@gmail.com

[2] College of Computing and Data Science (CCDS), Nanyang Technological University (NTU), Singapore
chunyun001@e.ntu.edu.sg,wslin@ntu.edu.sg

Abstract. Vision transformers (ViTs) have emerged as a promising alternative to convolutional neural networks (CNNs) for various image analysis tasks, offering comparable or superior performance. However, one significant drawback of ViTs is their resource-intensive nature, leading to increased memory footprint, computation complexity, and power consumption. To democratize this high-performance technology and make it more environmentally friendly, it is essential to compress ViT models, reducing their resource requirements while maintaining high performance. In this paper, we introduce a new block-structured pruning to address the resource-intensive issue for ViTs, offering a balanced trade-off between accuracy and hardware acceleration. Unlike unstructured pruning or channel-wise structured pruning, block pruning leverages the block-wise structure of linear layers, resulting in more efficient matrix multiplications. To optimize this pruning scheme, our paper proposes a novel hardware-aware learning objective that simultaneously maximizes speedup and minimizes power consumption during inference, tailored to the block sparsity structure. This objective eliminates the need for empirical look-up tables and focuses solely on reducing parametrized layer connections. Moreover, our paper provides a lightweight algorithm to achieve post-training pruning for ViTs, utilizing second-order Taylor approximation and empirical optimization to solve the proposed hardware-aware objective. Extensive experiments on ImageNet are conducted across various ViT architectures, including DeiT-B and DeiT-S, demonstrating competitive performance with other pruning methods and achieving a remarkable balance between accuracy preservation and power savings. Especially, we achieve up to **3.93**× and **1.79**× speedups on dedicated hardware and GPUs respectively for DeiT-B, and also observe an inference power reduction by **1.4**× on real-world GPUs. Code will be released soon.

Keywords: Network Pruning · Vision Transformers

* Equal contribution. ✉ Corresponding author.

1 Introductions

Recently, vision transformers (ViTs) have been an emerging string of research that greatly challenges the prevailing CNNs with on-par or even superior performance on various image analysis and understanding tasks such as classification [9, 13, 18, 22, 44], object detection [1, 3, 61], semantic segmentation [5, 35], etc., but completely without the convolution mechanism seen in the CNNs. Despite the success in the task performances, as pointed out by [54], one major drawback of the ViTs architecture is that the ViTs are much less resource-efficient than CNNs in terms of memory footprint, computation complexity and the eventual power consumption. To make the high-performance ViTs more environmental friendly and democratize the technology, it is necessary to compress the ViTs models and cut down the power consumption, so that they could be accessed by low-end computation devices with equal or comparable model performance.

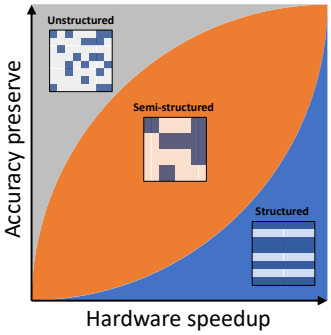

Fig. 1: Trade-offs of different sparsity schemes in terms of model accuracy and hardware acceleration.

Among different bifurcations of neural network compression, network pruning is an effective method that has shown success on CNNs, which prunes out redundant neurons or rules out computations in the networks. Previously on CNNs, some [20, 21, 31, 33, 40, 48, 50, 59] attempted *unstructured pruning* to the models which removes individual neurons from the layer weights; while others [36, 41] used *structured pruning* which removes channel-wise neurons. Comparing to unstructured pruning, the latter structured scheme has high data locality hence is more hardware-friendly [2] as it is easier to achieve accelerated computation by simply removing entire rows or columns in the weight matrices, it cause severer accuracy degradation due to the coarser pruning granularity making it a much more challenging pruning scheme.

Nevertheless, for transformer architectures consisting of mostly linear layers (matrix multiplication), block structured (semi-structured) pruning is a better trade off between accuracy and hardware acceleration, since the GEMM performs matrix multiplication in a block-by-block manner. Hence multiplication with block sparse matrices can achieve more speedup than unstructured ones under the same pruning ratio while still maintaining high accuracy. A summarized qualitative comparison among pruning schemes is listed in Fig. 1. Prior arts [30, 37] in NLP domain validated the block structured pruning on language models (BERT [11], MobileBERT [42], etc.), achieving more than 2× speedup with negligible performance drop. However, the other parts of their pruning scheme is rather out-dated, *e.g.* vanilla pruning criterion. Similar attempts are still scarce on ViTs for various vision tasks.

In this work, we propose a novel block-structured pruning approach for ViTs to prune the parameters in a block-based manner to achieve better trade-off

between accuracy and efficiency. We formulate the learning objective in a way that simultaneously maintains the accuracy of the pruned model and minimizes the number of the computational operations. A hardware-aware constraint is incorporated into the objective to boost the speedup and lower power consumption during inference stage. Moreover, we present a fast optimization method to solve the objective function by utilizing second-order Taylor approximation. After equivalent reformulation, such we are able to solve the objective very efficiently (quadratic to cubic complexity for empirical data collection against network size and linear time complexity for equation solving). To the best of our knowledge, this is the first paper that introduces the block-structured pruning scheme and present a hardware-aware post-training pruning approach for ViTs. The main contributions are summarized as below:

- We systematically formulate an optimal hardware-aware pruning objective for ViTs models under the block-structured pruning scheme, which directly optimizes both model accuracy and power consumption at the same time. The power consumption is fully estimated without constructing any empirical look-up tables (LUTs), making it a light-weight approach and does not require additional overheads for optimization. The proposed pruning scheme solely relies on reducing parametrized layer connections without manipulating skip configurations and token pruning.
- We then provide an efficient solution for the proposed hardware-aware objective function by second-order taylor approximation and present an empirical optimization method with only linear time complexity. With the weights being pruned, less parameters are required to read/write from off-chip memory to on-chip memory leading to memory reduction and significant power reduction.
- Extensive experiments demonstrate the effectiveness of our approach. Results on various deep ViTs architectures, including DeiT-B, DeiT-S, Swin-T, Swin-B, etc. show that our approach noticeably outperforms the state-of-the-arts while with up to $\mathbf{3.93}\times$ and $\mathbf{1.79}\times$ speedups on dedicated hardware and GPUs respectively for DeiT-B. Inference power reduction by $\mathbf{1.4}\times$ is also witnessed on real-world GPUs.

2 Related Works

2.1 Vision Transformers (ViTs)

Following the success of self-attention based transformer architecture in natural language processing [46], transformer based vision models have also been marching in image domain and being strong competitors against traditional CNNs in various scenes like object detection [3, 61], segmentation [5], etc. ViT [12] was the first attempt to introduce MHA (multi-head attention) architecture for image modality and surpassed the CNNs performance on image classification on large scale datasets. Later, DeiT [45] further boost the performance of raw ViTs with the same architecture but with token-based knowledge distillation to

enhance the representation learning. MAE [22] introduces a supervision technique to pretrain ViT encoder on masked image reconstruction pretext task and achieves state-of-the-art performance on ImageNet classification task. Swin Transformer [35] utilized shifted window to introduce inter-window information exchange and enhance local attention. Transformer-iN-Transformer (TNT) [19] aggregated both patch- and pixel-level representations by a nested self-attention within each transformer block.

2.2 Pruning on CNNs

CNNs pruning has been widely studied for decades. Large amount of pruning methods can be categorized into unstructured pruning, semi-structured pruning, structured (channel/filter-wise) pruning, etc., depending on the the level of sparsity.

Unstructured Pruning removes individual connections (neurons) from convolution kernels, which is the earliest established pruning scheme by the pioneer works [20, 21] that attempted unstructured pruning for LeNet and AlexNet. Following that, [15, 39, 40, 59] adopts magnitude-based or taylor-based criterion importance scores to threshold low-scored connections globally. [14, 17] leverage architectural heuristics to determine layerwise pruning rate. Several efforts [31,32,39] consider the influence of pruning on the model output to determine layerwise pruning rate. [24] assumed laplacian distribution of CNN weights to approximate output distortion. [48, 50] leverage rate-distortion theory to derive layer-wise pruning ratios that achieves optimal rate-distortion performance.

Structured Pruning or channel/filter-wise pruning scheme prunes the entire kernel in a Conv layer or a channel in fully connected layer at once. [36] used feature map importance as a proxy to determine removable channels. [23] took a regularization based strucrtural pruning method. [53] obtains channel-wise importance scores by propagating the score on the final response layer. [33] utilized rank informtation of feature maps to determine the prunable channels. [48] leveraged rate-distortion theory to prune the channels that lead to least model accuracy drop. [41] took first-order importance on channels and allocates sparsities by solving a knapsack problem on all channel importances in the whole network.

Semi-Structured Pruning is a less-explored approach that leverages sparsity pattern in between unstructured and structured pruning, where patterns such as block-sparsity in matmul can greatly benefit the realworld speedups by exploiting the nature of GPU calculation [30, 37]. With the sparsity pattern less agressive than sturctured pruning, the impact of removing neurons on the model accuracy is less than structured pruning. Nevertheless, semi-structured pruning is under-explored on the emerging ViTs, which are constructured with transformer encoder architecture with mostly fully connected layers.

2.3 Sparsity in ViTs

Witnessing the success of CNNs pruning, ViTs pruning is also receiving emerging interests. Compared to CNNs pruning, less efforts are devoted to pure weight pruning but more on pruning of tokens, MHA, etc. S^2ViTE [6] first proposed to prune out tokens as well as self-attention heads under structured pruning scheme with sparse training for ViTs. UVC [55] derived a hybrid optimization target that unifies structural pruning for ViT weights, tokens and skip configuration to achieve sparse training for ViTs. SPViT [27] only performed token pruning on attention heads but adopted latency constraint to maximize speedup on edge devices. [51] adopts Nvidia's Ampere 2:4 sparsity structure to achieve high speedup but required structural constraints to ensure a matching dimensions of qkv, feedforward and projection layers (head alignment) to search for subnetwork from larger ViT variants to match the latency of smaller ones. Unlike prior works [51, 54], our method focuses on pure weight pruning scheme and does not require heavy searching for the coordination of different compression schemes.

Some efforts [26, 47, 49, 56] sparsify the heavy self-attention by introducing sparse and local attention patterns for language models. [7] attempts on ViTs, but these sparse attention schemes still require training from scratch.

3 Methodologies

3.1 Preliminaries

Block-structured Pruning within layer. We targeted at block-structured pruning for all linear layer weights, which include any parametrized linear layers in the ViTs, such as qkv layers, feedforword and projection layers. Neurons in these weight matrices are grouped in 2-dimensional fixed-sized blocks as a unit for pruning. To decide which blocks need to be pruned, given a block structure (B_h, B_w), for each matrix $\boldsymbol{W} \in \mathbb{R}^{H \times W}$, we rank the blocks by the average of 1st order taylor expansion score of the neuron within each block. Mathematically, we first obtain the neuron score by the taylor expansion $\boldsymbol{S} = |\boldsymbol{W} \cdot \nabla_{\boldsymbol{W}} f|$ similar to [38], then perform a 2D average pooling to obtain a score for each block $\boldsymbol{S'} \in \mathbb{R}_*^{H/B_h \times W/B_w}$ ($\mathbb{R}_*$ is non-negative real value set). Then given a pruning ratio for each layer, we can rank the blocks by their scores and eliminate the bottom ranked ones. The right most part of Fig. 2 visualizes the block-structure patterns realistically generated from ViTs. The above pruning scheme can be formulated as $\widetilde{\boldsymbol{W}}_{i,j} = \boldsymbol{W}_{i,j} \odot \boldsymbol{M}_\alpha(\boldsymbol{S'})_{\lceil \frac{i}{B_h} \rceil, \lceil \frac{j}{B_w} \rceil}$, where $\boldsymbol{M}_\alpha(\boldsymbol{S'})$ is the binary mask generated from the previous block-wise score matrix under the pruning ratio α.

Pruning scheme of ViTs. Unlike prior arts, the scope of this work is only eliminating model parameters to reduce computation, without considering other aspects of ViTs like token number and token size and transformer block skipping [6, 27, 55].

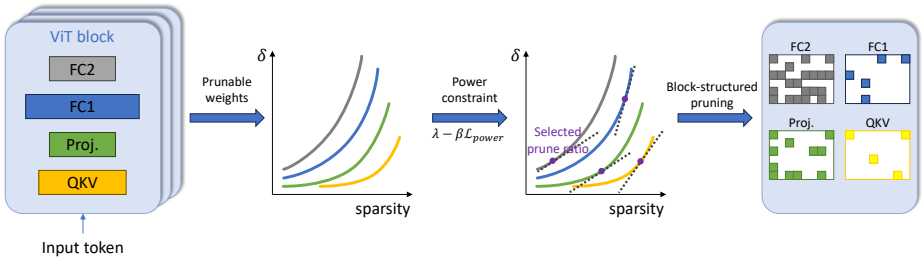

Fig. 2: Illustration of the proposed Low Power Semi-structured pruning method. Widths of different layers within ViT block visualizes the computation complexities (FLOPs) of single layer. We first extract all layers with prunable weights in the pre-trained ViTs, then we obtain the empirical curves δ-vs-sparsity as described in Eq. 12. We further calculate the layer specific target slope λ_i according to its contribution to the power consumption and select the layer-wise pruning ratios when the target slopes are tangential to the curves. Finally we prune the layer weights given their pruning ratios in block-structured sparsity, and finally finetune the pruned ViTs. The rightmost of the diagram is an example of the block-sparsity patterns when block sizes for both dimensions are the same, but they don't have to be the same as in the experiment section.

We further adopt a basic assumption for the weight perturbation $\Delta \boldsymbol{W} = \widetilde{\boldsymbol{W}} - \boldsymbol{W}$ caused by a typical pruning operation to the weight:

Assumption 1 ***I.i.d. weight perturbation across layers*** *[58]: This means the joint distribution is zero-meaned:*

$$\forall 0 < i \neq j < L, E(\Delta \boldsymbol{W}^{(i)} \Delta \boldsymbol{W}^{(j)}) = E(\Delta \boldsymbol{W}^{(i)}) E(\Delta \boldsymbol{W}^{(j)}) = 0, \tag{1}$$

and also zero co-variance: $E(\|\Delta \boldsymbol{W}^{(i)} \Delta \boldsymbol{W}^{(j)}\|^2) = 0$.

3.2 Hardware-aware pruning objective

Since layers may contribute differently to the model performance [16], various criteria have been proposed to allocate layerwise sparsity given a total budget [24,31,32,50]. However, most existing pruning objectives can be summarized as minimizing the model output accuracy under computation constraint, without explicitly taking into account the actual power consumption and speedup. In contrast, our compression pruning objective directly optimizes the power consumption to achieve certain computation reduction target (FLOPs). Specifically, given a neural network f of l layers and its parameter set $\boldsymbol{W}^{(1:l)} = (\boldsymbol{W}^{(1)}, ..., \boldsymbol{W}^{(l)})$, where $\boldsymbol{W}^{(i)}$ is the weights in layer i, pruning parameters in the f will give a new parameter set $\widetilde{\boldsymbol{W}}^{(1:l)}$. We view the impact of pruning as the distance between the network outputs $f(x; \boldsymbol{W}^{(1:l)})$ and $f(x; \widetilde{\boldsymbol{W}}^{(1:l)})$.

Hence our learning objective is as follows:

$$\begin{aligned} &\min \|f(x; \boldsymbol{W}^{(1:l)}) - f(x; \widetilde{\boldsymbol{W}}^{(1:l)})\|^2 + \beta \mathcal{L}_{power}(f(\widetilde{\boldsymbol{W}}^{(1:l)})), \\ &s.t. \ \frac{\text{FLOPs}(f(\widetilde{\boldsymbol{W}}^{(1:l)})}{\text{FLOPs}(f(\boldsymbol{W}^{(1:l)})} \leq R, \end{aligned} \tag{2}$$

which jointly minimize the output distortion caused by pruning (first term) as well as the estimated power consumption $\mathcal{L}_{power}(f(\widetilde{\boldsymbol{W}}^{(1:l)}))$, under a certain FLOPs reduction target R.

3.3 Second-order Approximation of Output Distortion

To solve the pruning objective, we break down the first term related to the output distortion. We first expand the output distortion $f(x;\boldsymbol{W}^{(1:l)})-f(x;\widetilde{\boldsymbol{W}}^{(1:l)})$ using second-order taylor expansion: (omit the superscript $(1:l)$ for visual clarity from now)

$$f(x;\boldsymbol{W})-f(x;\widetilde{\boldsymbol{W}})=\sum_{i=1}^{l}\nabla_{\boldsymbol{W}^{(i)}}^{\top}f\Delta\boldsymbol{W}^{(i)}+\frac{1}{2}\Delta\boldsymbol{W}^{(i)\top}\boldsymbol{H}_i\Delta\boldsymbol{W}^{(i)}, \tag{3}$$

where $\boldsymbol{H}_i$ is the hessian matrix of the i-th layer weight.

Then consider the expectation of the squared L2 norm in the objective Eq. 2, which can be rewritten as the vector inner-product form:

$$\begin{aligned}&E(\|f(x;\boldsymbol{W})-f(x;\widetilde{\boldsymbol{W}})\|^2)=E\left[(f(x;\boldsymbol{W})-f(x;\widetilde{\boldsymbol{W}})^{\top}(f(x;\boldsymbol{W})-f(x;\widetilde{\boldsymbol{W}})\right]\\&=\sum_{i,j=1}^{l}E\left[\left(\nabla_{\boldsymbol{W}^{(i)}}^{\top}f\Delta\boldsymbol{W}^{(i)}+\frac{1}{2}\Delta\boldsymbol{W}^{(i)\top}\boldsymbol{H}_i\Delta\boldsymbol{W}^{(i)}\right)^{\top}\left(\nabla_{\boldsymbol{W}^{(j)}}^{\top}f\Delta\boldsymbol{W}^{(j)}+\frac{1}{2}\Delta\boldsymbol{W}^{(j)\top}\boldsymbol{H}_j\Delta\boldsymbol{W}^{(j)}\right)\right].\end{aligned} \tag{4}$$

When we further expand the inner-product term, the cross-term for each pair of different layer $1\le i\neq j\le l$ is:

$$\begin{aligned}&E\left[\Delta\boldsymbol{W}^{(i)\top}\nabla_{\boldsymbol{W}^{(i)}}f\nabla_{\boldsymbol{W}^{(j)}}^{\top}f\Delta\boldsymbol{W}^{(j)}\right]+E\left[\frac{1}{2}\Delta\boldsymbol{W}^{(i)}\Delta\boldsymbol{W}^{(i)\top}\boldsymbol{H}_i^{\top}\nabla_{\boldsymbol{W}^{(j)}}^{\top}f\Delta\boldsymbol{W}^{(j)}\right]+\\&E\left[\frac{1}{2}\Delta\boldsymbol{W}^{(i)\top}\nabla_{\boldsymbol{W}^{(i)}}f\Delta\boldsymbol{W}^{(j)\top}\boldsymbol{H}_j\Delta\boldsymbol{W}^{(j)}\right]+E\left[\frac{1}{4}\Delta\boldsymbol{W}^{(i)}\Delta\boldsymbol{W}^{(i)\top}\boldsymbol{H}_i^{\top}\Delta\boldsymbol{W}^{(j)\top}\boldsymbol{H}_j\Delta\boldsymbol{W}^{(j)}\right].\end{aligned} \tag{5}$$

When we dicuss the influence of the random variable $\Delta\boldsymbol{W}$, the first-order and second-order derivatives $\nabla_{\boldsymbol{W}}f$ and $\boldsymbol{H}$ can be regarded as constants and therefore can be moved out of expectation. Also vector transpose is agnostic inside expectation. So Eq. 5 becomes

$$\begin{aligned}&\nabla_{\boldsymbol{W}^{(i)}}f\nabla_{\boldsymbol{W}^{(j)}}^{\top}fE(\Delta\boldsymbol{W}^{(i)\top}\Delta\boldsymbol{W}^{(j)})+\frac{1}{2}\boldsymbol{H}_i^{\top}\nabla_{\boldsymbol{W}^{(j)}}^{\top}fE(\Delta\boldsymbol{W}^{(i)}\Delta\boldsymbol{W}^{(i)\top}\Delta\boldsymbol{W}^{(j)})+\\&\frac{1}{2}\nabla_{\boldsymbol{W}^{(i)}}f\boldsymbol{H}_jE(\Delta\boldsymbol{W}^{(i)\top}\Delta\boldsymbol{W}^{(j)\top}\Delta\boldsymbol{W}^{(j)})+\frac{1}{4}\boldsymbol{H}_i^{\top}\boldsymbol{H}_jE(\|\Delta\boldsymbol{W}^{(i)\top}\Delta\boldsymbol{W}^{(j)}\|^2).\end{aligned} \tag{6}$$

Using Assumption 1, we can find that the above 4 cross-terms also equal to zero . Therefore the expectation Eq. 4 results in only intra-layer terms:

$$E(\|f(x;\boldsymbol{W})-f(x;\widetilde{\boldsymbol{W}})\|^2)=\sum_{i=1}^{l}E\left(\left\|\nabla_{\boldsymbol{W}^{(i)}}^{\top}f\Delta\boldsymbol{W}^{(i)}+\frac{1}{2}\Delta\boldsymbol{W}^{(i)\top}\boldsymbol{H}_i\Delta\boldsymbol{W}^{(i)}\right\|^2\right). \tag{7}$$

We empirically find $E(\boldsymbol{W}^{(i)\top}\boldsymbol{W}^{(i)}\boldsymbol{W}^{(j)})=0$ holds on top of $E(\boldsymbol{W}^{(i)}\boldsymbol{W}^{(j)})=0$.

3.4 Power consumption under Block-structured Pruning

As the majority of the power consumption of network inference is attributed to the matrix multiplication operation, the network power consumption can be estimated by summing individual power cost of block-sparse matrix multiplication of each linear layers. Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, typically input tensor, to be multiplied with the block-sparse weight matrix $\boldsymbol{B} \in \mathbb{R}^{N\times K}$ with block-structure of (B_n, B_k) and α-percentage of blocks pruned out. When using a block-sparse GEMM configured with the kernel grid size of B_m on M-dimension, the power consumption of the block-sparse matmul can be estimated as

$$P = p_m \frac{M}{B_m} \left\lceil (1-\alpha) \frac{N}{B_n} \frac{K}{B_k} \right\rceil, \tag{8}$$

where p_m is the power cost of individual within-block matmul. Therefore, the second term in Eq. 2 can be obtained by adding up the power consumption of the network of all layers:

$$\beta \mathcal{L}_{power} = \beta p_m \sum_{i=1}^{l} \frac{M_i}{B_m} \left\lceil (1-\alpha_i) \frac{N_i}{B_n} \frac{K_i}{B_k} \right\rceil, \tag{9}$$

where p_m and B_m can be absorbed into the weight coefficient β because they only depends on hardware parameters and GEMM configuration which is unified across layers.

Final Objective. Combining Eq. 7 and Eq. 9, the final objective can be reformulated as:

$$\begin{aligned} \min \ & \sum_{i=1}^{l} E\left(\left\| \nabla^{\top}_{\boldsymbol{W}^{(i)}} f \Delta \boldsymbol{W}^{(i)} + \frac{1}{2} \Delta \boldsymbol{W}^{(i)\top} \boldsymbol{H}_i \Delta \boldsymbol{W}^{(i)} \right\|^2 \right) + \beta \sum_{i=1}^{l} M_i \left\lceil (1-\alpha_i) \frac{N_i}{B_n} \frac{K_i}{B_k} \right\rceil \\ s.t. \ & \frac{\text{FLOPs}(f(\widetilde{\boldsymbol{W}}^{(1:l)})}{\text{FLOPs}(f(\boldsymbol{W}^{(1:l)})} \leq R. \end{aligned} \tag{10}$$

3.5 Finding Solution to Pruning Objective

At this point, we can further solve the optimization problem Eq. 10 on the layer-wise pruning ratio set $\{\alpha_i \mid 1 \leq i \leq l\}$ by applying lagrangian formulation [48,50]

$$\frac{\partial}{\partial \alpha_i} \left(\left\| \nabla^{\top}_{\boldsymbol{W}^{(i)}} f \Delta \boldsymbol{W}^{(i)} + \frac{1}{2} \Delta \boldsymbol{W}^{(i)\top} \boldsymbol{H}_i \Delta \boldsymbol{W}^{(i)} \right\|^2 + \beta M_i \left\lceil (1-\alpha_i) \frac{N_i}{B_n} \frac{K_i}{B_k} \right\rceil \right) = \lambda. \tag{11}$$

In practice we can get rid of the ceiling function in Eq. 11 and therfore:

$$\frac{\partial}{\partial \alpha_i} \left(\left\| \nabla^{\top}_{\boldsymbol{W}^{(i)}} f \Delta \boldsymbol{W}^{(i)} + \frac{1}{2} \Delta \boldsymbol{W}^{(i)\top} \boldsymbol{H}_i \Delta \boldsymbol{W}^{(i)} \right\|^2 \right) = \lambda_i = \lambda + \beta \frac{M_i N_i K_i}{B_n B_k}, \tag{12}$$

which will give a continuous $\alpha_i \in [0, 1]$ compared to the original solution with the ceiling, but in practice since the number of blocks within a weight tensor is limited the pruning ratio α_i is to be rounded to a discrete value anyway. Solving Eq. 12 will need to collect empirical curves for all layers (pruning ratio α_i against the taylor second-order term $\delta_i = \nabla^{\top}_{\boldsymbol{W}^{(i)}} f \Delta \boldsymbol{W}^{(i)} + \frac{1}{2} \Delta \boldsymbol{W}^{(i)\top} \boldsymbol{H}_i \Delta \boldsymbol{W}^{(i)}$). By setting a specific λ, we can solve Eq. 12 individually for each layer by searching for a α_i that let the equality holds. The final solution of pruning ratios can be

obtained by traversing λ that returns a pruned network closest to the constraint R.

One *key insight* that one can derive from the optimization solution Eq. 12 is that by controlling the weight β, the power consumption are explicitly incorporated in the optimization process in the form of altering the target slope for the partial derivative of the curve $\frac{\partial \delta_i(\alpha_k)}{\partial \alpha_k}$, which represents how intensely pruning one layer affects the final model accuracy (output distortion). In this way, we achieve direct tradeoff between model accuracy and power consumption.

3.6 Empirical Complexity

Hessian approximation. For empirical networks, we approximate the hessian matrix $\boldsymbol{H}_i$ using *empirical fischer* [29]:

$$\boldsymbol{H}_i = \boldsymbol{H}_{\mathcal{L}}(\boldsymbol{W}^{(i)}) \approx \hat{\boldsymbol{F}}(\boldsymbol{W}^{(i)}) = \kappa \boldsymbol{I}_d + \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{W}^{(i)}} f_n \nabla^{\top}_{\boldsymbol{W}^{(i)}} f_n. \tag{13}$$

In order to obtain empirical curves $\frac{\partial \delta_i(\alpha_k)}{\partial \alpha_k}$ on a calibration set, one is possible to traverse different pruning ratio (*e.g.* in practice $\alpha_k = \frac{k+1}{K}, 0 < k < K$) and caluculate the corresponding $\delta_i(\alpha_k)$ for all $0 < k < K$. However in such case, even with the approximated hessian, the curve generation for each layer is still very expensive at the complexity of $O(NKD_i^4)$, where K is the number of possible pruning ratio selections and $D_i = N_i K_i$ is the dimension of weight in i-th layer. This poses challenge to make the proposed method efficient enough to enjoy the benefits of sparse network. We notice that the derivative $\nabla_{\boldsymbol{W}_i}$ is constant to the change of pruning ratio which let us to reuse the hessian matrix for all pruning ratio, which drops the complexity to $O((N+K)D_i^2 + KD_i^4)$. However, the existence of the biquadratic complexity makes it still too expensive. We further notice that when pruning ratio move up slightly, only a partition of the weight vector is pruned out from $\widetilde{\boldsymbol{W}}_i$. Therefore we can select a subvector $\mathrm{d}\Delta\boldsymbol{W}_i(\alpha_k) = \Delta\boldsymbol{W}_i(\alpha_k) - \Delta\boldsymbol{W}_i(\alpha_{k-1})$ each time when pruning ratio increases from α_{k-1} to α_k and update the $\delta_i(\alpha_k)$ from $\delta_i(\alpha_{k-1})$ by the following rule:

$$\delta_i(\alpha_k) - \delta_i(\alpha_{k-1}) = \nabla^{\top\prime}_{\boldsymbol{W}^{(i)}} f \mathrm{d}\Delta\boldsymbol{W}_i(\alpha_k) + \left(\frac{1}{2}\mathrm{d}\Delta\boldsymbol{W}_i(\alpha_k) + \Delta\boldsymbol{W}_i(\alpha_{k-1})\right)^{\top} \boldsymbol{H}'_i \mathrm{d}\Delta\boldsymbol{W}_i(\alpha_k). \tag{14}$$

Denote the dimension of the subvector $\mathrm{d}\Delta\boldsymbol{W}_i(\alpha_k)$ as $d_i(k) \ll D_i$ equals the number of values changes from $\Delta\boldsymbol{W}_i(\alpha_{k-1})$ to $\Delta\boldsymbol{W}_i(\alpha_k)$, the multiplication calculation in Eq. 14 can be operated at lower dimensions, where $\nabla^{\top\prime}_{\boldsymbol{W}^{(i)}} f \in \mathbb{R}^{d_i(k)}, \boldsymbol{H}'_i \in \mathbb{R}^{D_i \times d_i(k)}$ are subvector and submatrix indexed from the original ones. At $k = 1$, $\alpha_k = 0$ *i.e.* there is no pruning at all which guarantees $\delta_i(\alpha_1) = 0$. Since α_k increases linearly, the $d_i(k) \approx \frac{D_i}{K}$, therefore, the complexity is around $O(\frac{N}{2}D_i^2)$. Hence the total computation complexity for calculating δ for all l layers is around $O(\frac{1}{2}\sum_i^l D_i^2)$, which is far less than the original complexity.

To this end, we presented a hardware-aware pruning criterion that explicitly accounts for the power consumption of the block-structured sparse model inference. The block-structured pruning scheme enables the obtained sparse network to achieve real-world acceleration on hardware while optimally preserving the network accuracy. The algorithm is extremely efficient to obtain a sparse ViT.

Table 1: Comparisons with state-of-the-art ViTs pruning methods. We label the sparsity schemes the baselines exploit for a fair comparison, where **S**: Structured Sparsity, **U**: Unstructured Sparsity, **B**: Blocksparse, **T**: Token selection(gating), **A**: Attention. For LPViT (Ours), we report the best results among different blocksize configurations.

Model	Method	Sparsity Scheme	FLOPs(G)	FLOPs remained (%)	Top-1 Acc (%)
Deit-Small	Dense		4.6	100	79.8
	Uniform	**U**	2.3	50	77.17 (-2.63)
	SCOP	**S**	2.6	56.4	77.5 (-2.3)
	S^2ViTE	**S+T**	3.14	68.36	79.22 (-0.58)
	UVC	**S+A+T**	2.32	50.41	78.82 (-0.98)
	LPViT	**B**	2.3	50	**80.69**(+**0.89**)
Deit-Base	Dense		17.6	100	81.8
	S^2ViTE	**S+T**	11.87	66.87	82.22 (+0.42)
	VTP	**U**	10	56.8	80.7 (-1.1)
	UVC	**S+A+T**	8	45.5	80.57 (-1.23)
	LPViT	**B**	8.8	50	**80.81**(−**0.99**)
	LPViT	**B**	7.92	45	80.55 (-1.25)
	LPViT	**B+A+T**	5.10	29	**81.03**(−**0.77**)
Swin-Base	Dense		15.4	100	83.5
	LPViT	**B**	11.24	73	81.73 (-1.77)
Swin-Tiny	Dense		4.5	100	81.2
	X-Pruner	**S+A**	3.19	71.1	78.55 (-2.65)
	LPViT	**B**	3.47	77	80.0(−1.2)
	LPViT	**B**	3.19	71.1	**79.24**(−**1.96**)
	LPViT	**B+A**	2.72	60.47	**80.0**(−**1.2**)

4 Experiments

4.1 Datasets and Benchmarks

We extensively evaluate our ViTs pruning method for image classification task as well as a downstream image segmentation task. For classification task, we conduct experiments mainly on Deit [45] model as well as Swin Transformers [35] on ImageNet dataset [28]. We adopt the same training settings as in UVC [54] for the finetuning of ViTs. For image segmentation, we test on the transferred performance of DeiT-Base backbone in SETR [57] model on Cityscapes dataset [10]. More implementation details can be found in supplementary materials.

Baseline methods. For the following experiments, we followed the UVC [54] comparison settings and compare ourselves to the previous ViTs compression methods that at least involves model weights pruning, as well as hybrid methods, including SCOP [43], VTP [60], S^2ViTE [6], X-Pruner [52] and UVC [54] itself. We also include a uniform pruning result for Deit-Base where it fixes a uniform pruning ratio for all layers.

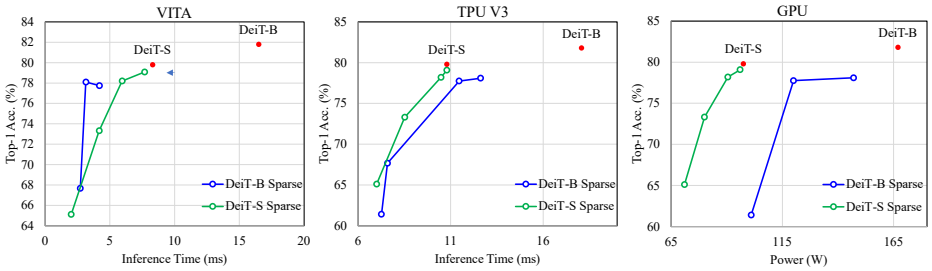

Fig. 3: Inference overhead and power reductions on hardware platforms.

4.2 Main results

As presented in Tab. 1, we first notice that our result on Deit-Small achieves loss-less, and even higher than dense model performance by 0.89, with roughly the same FLOPs, surpassing existing baselines by a **large margin**. On larger architectures like DeiT-Base, where our method displays less prominent improvement but still on-par performance on the Top-1 accuracy of 80.81 with 50% FLOPs remaining and 80.55 with around 45% FLOPs. This is an intuitive observation since coarser pruning patterns like structural pruning would hurt the performance of smaller models more than larger model with a lot more redundant weights, and that is also where smaller structures such as the proposed block-sparsity pattern will retain more performance while still ensure speedup compared to unstructured pruning. On DeiT-Small, we beat all existing hybrid methods that leverage patch-slimming or token selections. We remain competitive on larger model DeiT-Base, while we notice S^2ViTE cannot achieve comparable FLOPs reduction to us. On non-global attention transformers such as Swin Transformer, LPViT remains competitive compared to other ViT pruning methods, achieving only 1.96% loss on Swin-Tiny with FLOPs 71.7%. We also report a mere 1.77 drop on Swin-Base. We show less improvement on DeiT-B despite more parameters than DeiT-S. Although seems to be counterintuitive, the reason can be that larger models are easier to get overfitted at finetuning stage so that accuracy cannot be fully recovered. This can also be attributed to the regularizing effects of pruning, benefitting to the OOB data performance, and has been discovered in many prior papers [34]. Meanwhile, LPViT is a generic framework which also supports integration with other sparsity. When combined with attention and token pruning, LPViT further improves accuracy with even less FLOPs for DeiT-Base and Swin-Tiny.

4.3 Hardware Performance Benchmarks

We evaluate the inference efficiencies of our pruned models in terms of speedup and power consumption reductions on three types of hardware platforms and summarize the results in Tab. 2. We first simulated the statistics of ViTs on a RISC-V platform ViTA [4], which supports GEMMs in various blocksizes, enabling us to fulfill the theoretical speedups of blocksparse models. Our approach

Table 2: Frontier speedup Results on various hardware platforms and Power Reduction on A-100 GPU.

Model	FLOPs Remained (%)	Blocksize	Top-1 Accuracy	VITA	TPU-V3	A-100	Power (W)
DeiT-Base	100	-	81.8	-	-	-	167
	25.8	32×32	77.75	**3.93**×	1.57×	1.75×	**120**(**71.8**%)
	38.5	64×32	78.1	2.61×	1.43×	1.79×	147(88%)
DeiT-Small	100	-	79.8	-	-	-	195
	71.3	16×32	78.2	**1.4**×	1.03×	1.2×	**181**(**92.8**%)
	75	16×16	80.65	1.33×	1.09×	1.14×	/

Table 3: Ablation studies on the Power consumption constraint on the pruning result. We compare between the results with the power constraint (main results) and without (by setting $\beta = 0$).

Method	Top-1 Acc (%)	Params remained (%)	FLOPs remained (%)
Deit-Base-BK32BN32			
w/ Power constraint	80.81	73.3	52.5
w/o Power constraint	77.75	26.9	55.6
Deit-Base-BK32BN64			
w/ Power constraint	80.71	72.8	50
w/o Power constraint	61.42	49	49.7

thereby obtains as high as **3.93**× and **1.4**× speedups for DeiT-Base and DeiT-Small respectively. We then perform simulation on a high-bandwidth platform Google TPU-V3 [25]. Since TPU V3 only offers 1 type of MAC with block size fixed at 128×128, we expect less speedup than on RISC-V. Nevertheless, we still see positive **1.57**× speedup for DeiT-Base. Finally, we deploy on NVIDIA A100 40GB GPU with CUDA 11.8 and evaluate the end-to-end inference time and the runtime power consumption, and observe a power reduction up to 71.8% on DeiT-Base. Power consumption is measured by averaging `nvidia-smi`'s power meter over an adequate time period and subtracting the idle power consumption.

Fig. 3 shows the results when pruning the models to different sizes, *i.e.* the inference time and accuracy at different pruning levels. Specifically, each dot represents one result for a specific pruning configuration. In Fig. 3, we observed that on VITA, DeiT-B obtains better trade-off than DeiT-S, while on TPU and GPU, DeiT-S performs better. This shows that the models may perform differently on different hardware platforms.

4.4 Ablation Study and Discussions

Beyond the main results, we also attempt to discover how each creative parts in our proposed pruning scheme contribute to the final results, *e.g.* the essential objective constraint regulating the power consumption and the block-sparsity structure, and answering the important questions such as why does the power

Table 4: Ablation study of different Block shape configurations on the pruning result.

Model	Block shape (BK × BN)	Sparsity (%)	Top-1 Acc (%)	FLOPs remained (%)
Deit-Small	16 × 16	92.2	80.69	50
	32 × 16	91	79.09	50
	16 × 32	71.37	78.2	50
	32 × 32	49	73.32	50
Deit-Base	32 × 32	72.84	80.81	52.5
	64 × 32	33.93	80.05	50
	32 × 64	16.99	80.71	50
	64 × 64	73.34	79.52	50.2

constraint benefits the performance. We present the detailed ablation studies in the Tab. 3 and Tab. 4.

Power constraint. We compared the behaviors of our pruning objective with and without the second-term power consumption in Eq. 10. As shown in Tab. 3, we notice that under the same level of FLOPs reduction rate of 50% , our final pruning scheme (with power constraint) constantly gives significant higher results under different block shapes. This is an inspiring phenomenon since the power constraint are not designed to facilitate model accuracy at the first place. By inspecting the model sparsity, we learn that the proposed power constraint looks for layers with larger matmul dimensions to allocate more pruning quota to achieve most reduction in computation, and normally larger layers have more parameter redundancy. Therefore, this pruning ratio allocation actually cooperates with the main objective to minimize output distortion.

Block structure configurations. How well our optimization scheme adapts to different block size configurations is crucial to generalize on different hardware platforms with different levels of parallelism. Therefore, we conducted an ablation studies varying different block shapes combinations as listed in Tab. 4. Firstly, we observe that smaller block sizes preserve more model accuracy. Secondly, we notice that smaller networks are more sensitive to the change of block shapes, where different block sizes results in drastic change to the resulting number of parameters left in the networks.

Layerwise-sparsity Allocation. We visualize the optimization results for each individual settings in Tab. 4 as below. we notice some interesting observations. First, LPViT preserved more connections in the last transformer blocks including the classification head on both DeiT-B and DeiT-S on different pruning ratios, and the classification head is almost kept unpruned in all cases. Second, on both DeiT-B and DeiT-S, we notice the projection layers after MHA in particular are always getting high pruning ratio, showing that projection layers in ViTs have more redundancies and effect the model performance the least. The above patterns are all automatically learned from our second-order pruning layer-wise sparsity allocation algorithm, showing the effectiveness of our method.

Self-attention Maps. As shown in Fig. **??**, we followed the same self-attention maps visualization process adopted in [6, 8] to show potential influence of the

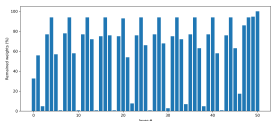

Fig. 4: Layerwise sparsity for DeiT-Base BK64BN64.

Table 5: Segmentation results on Cityscapes validadtion dataset.

Backbone	Method	FLOPs (G)	mIoU
DeiT-Base/384	Dense	17.6	78.66
DeiT-Base/384	LPViT	8.8	77.39

pruning on the attention behaviors in transformer multi-head attention. We observe that our LPViT block-sparsity pruning scheme displays a coarse and discretized pattern in a lot of attention heads across transformer blocks, accompanied by some completely inactive attention heads which allows us to further speedup inference by directly skiping those heads. Compared to structural pruning scheme, our semi-structured scheme allows middle states between blank attention and the delicate pattern in dense model, preserving more attention information which is curcial to the model accuracy. On LPViT-DeiT-Base (FLOPs 45%), the last two attention layers have no active attention heads. As a result, the entire blocks can be discarded in calculation, which could possibly bring the reported FLOPs reduction even more.

Transfer Learning to Downstream Tasks. To evaluate the generalizability of LPViT on downstream tasks, we further evaluate on transferred learning performance of our method on the downstream Cityscapes [10] segmentation task. We first pretrained a DeiT-B/384 pruned by 50% FLOPS on imagenet for 27 epochs and further use it as a backbone in a recent segmentation model SETR [57] and train on Cityscapes dataset. Tab. 5 compares the val mIoU of the pruned backbone and the original performance. We only observe a performance degration of mere 1.27 mIoU. More detailed results can be found in supplementary material.

5 Conclusions

In this work, we presented a novel ViTs weight pruning algorithm designed to reduce energy consumption during inference. Leveraging the linear layer-centric structure of the ViT architecture, we introduced a semi-structured pruning scheme to balance finetuning stability and hardware efficiency. Our algorithm is very efficient despite employing a hessian-based pruning criterion. Experimental results on various ViTs on ImageNet showcase the method's ability to identify optimal pruning solutions, maximizing accuracy for block-sparse models. Additionally, we illustrated the dual benefits of our proposed power-aware pruning objective, enhancing both software accuracy and hardware acceleration.

References

1. Amini, A., Periyasamy, A.S., Behnke, S.: T6d-direct: Transformers for multi-object 6d pose direct regression. In: DAGM German Conference on Pattern Recognition. pp. 530–544. Springer (2021)

2. Buluc, A., Gilbert, J.R.: Challenges and advances in parallel sparse matrix-matrix multiplication. In: 2008 37th International Conference on Parallel Processing. pp. 503–510. IEEE (2008)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. Computer Vision–ECCV 2020 pp. 213–229 (2020)
4. Chen, C., Li, L., Aly, M.M.S.: Vita: A highly efficient dataflow and architecture for vision transformers. In: 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE (2024)
5. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., Gao, W.: Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12299–12310 (2021)
6. Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., Wang, Z.: Chasing sparsity in vision transformers: An end-to-end exploration. Advances in Neural Information Processing Systems **34**, 19974–19988 (2021)
7. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)
8. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. In: International Conference on Learning Representations (2019)
9. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HJlnC1rKPB
10. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
14. Evci, U., Gale, T., Menick, J., Castro, P.S., Elsen, E.: Rigging the lottery: Making all tickets winners. In: International Conference on Machine Learning. pp. 2943–2952. PMLR (2020)
15. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=rJl-b3RcF7
16. Frankle, J., Dziugaite, G.K., Roy, D., Carbin, M.: Pruning neural networks at initialization: Why are we missing the mark? In: International Conference on Learning Representations (2020)
17. Gale, T., Elsen, E., Hooker, S.: The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574 (2019)

18. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. Advances in Neural Information Processing Systems **34**, 15908–15919 (2021)
19. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. Advances in Neural Information Processing Systems **34**, 15908–15919 (2021)
20. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
21. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)
22. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
23. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision. pp. 1389–1397 (2017)
24. Isik, B., Weissman, T., No, A.: An information-theoretic justification for model pruning. In: International Conference on Artificial Intelligence and Statistics. pp. 3821–3846. PMLR (2022)
25. Jouppi, N.P., Yoon, D.H., Kurian, G., Li, S., Patil, N., Laudon, J., Young, C., Patterson, D.: A domain-specific supercomputer for training deep neural networks. Communications of the ACM **63**(7), 67–78 (2020)
26. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: The efficient transformer. In: International Conference on Learning Representations (2019)
27. Kong, Z., Dong, P., Ma, X., Meng, X., Niu, W., Sun, M., Shen, X., Yuan, G., Ren, B., Tang, H., et al.: Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In: European Conference on Computer Vision. pp. 620–640. Springer (2022)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
29. Kurtic, E., Campos, D., Nguyen, T., Frantar, E., Kurtz, M., Fineran, B., Goin, M., Alistarh, D.: The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 4163–4181 (2022)
30. Lagunas, F., Charlaix, E., Sanh, V., Rush, A.M.: Block pruning for faster transformers. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 10619–10629 (2021)
31. Lee, J., Park, S., Mo, S., Ahn, S., Shin, J.: Layer-adaptive sparsity for the magnitude-based pruning. In: International Conference on Learning Representations (2020)
32. Lee, N., Ajanthan, T., Torr, P.: Snip: single-shot network pruning based on connection sensitivity. In: International Conference on Learning Representations. Open Review (2019)
33. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1529–1538 (2020)
34. Liu, J., Niu, L., Yuan, Z., Yang, D., Wang, X., Liu, W.: Pd-quant: Post-training quantization based on prediction difference metric. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24427–24437 (2023)

35. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
36. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision. pp. 5058–5066 (2017)
37. Mao, J., Yang, H., Li, A., Li, H., Chen, Y.: Tprune: Efficient transformer pruning for mobile devices. ACM Transactions on Cyber-Physical Systems **5**(3), 1–22 (2021)
38. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11264–11272 (2019)
39. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: International Conference on Learning Representations (2016)
40. Morcos, A., Yu, H., Paganini, M., Tian, Y.: One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. Advances in neural information processing systems **32** (2019)
41. Shen, M., Yin, H., Molchanov, P., Mao, L., Liu, J., Alvarez, J.M.: Structural pruning via latency-saliency knapsack. Advances in Neural Information Processing Systems **35**, 12894–12908 (2022)
42. Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., Zhou, D.: Mobilebert: a compact task-agnostic bert for resource-limited devices. arXiv preprint arXiv:2004.02984 (2020)
43. Tang, Y., Wang, Y., Xu, Y., Tao, D., Xu, C., Xu, C., SCOP, C.X.: Scientific control for reliable neural network pruning. Neural Information Processing Systems (NeurIPS) **1**(2), 7 (2020)
44. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
45. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
46. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
47. Wang, H., Zhang, Z., Han, S.: Spatten: Efficient sparse attention architecture with cascade token and head pruning. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). pp. 97–110. IEEE (2021)
48. Wang, Z., Lin, J., Geng, X., Aly, M.M.S., Chandrasekhar, V.: Rdo-q: Extremely fine-grained channel-wise quantization via rate-distortion optimization. In: European Conference on Computer Vision. pp. 157–172. Springer (2022)
49. Wu, Z., Liu, Z., Lin, J., Lin, Y., Han, S.: Lite transformer with long-short range attention. In: International Conference on Learning Representations (2019)
50. Xu, K., Wang, Z., Geng, X., Wu, M., Li, X., Lin, W.: Efficient joint optimization of layer-adaptive weight pruning in deep neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17447–17457 (2023)
51. Yang, H., Yin, H., Shen, M., Molchanov, P., Li, H., Kautz, J.: Global vision transformer pruning with hessian-aware saliency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18547–18557 (2023)

52. Yu, L., Xiang, W.: X-pruner: explainable pruning for vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24355–24363 (2023)
53. Yu, R., Li, A., Chen, C.F., Lai, J.H., Morariu, V.I., Han, X., Gao, M., Lin, C.Y., Davis, L.S.: Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9194–9203 (2018)
54. Yu, S., Chen, T., Shen, J., Yuan, H., Tan, J., Yang, S., Liu, J., Wang, Z.: Unified visual transformer compression. In: International Conference on Learning Representations (2021)
55. Yu, S., Chen, T., Shen, J., Yuan, H., Tan, J., Yang, S., Liu, J., Wang, Z.: Unified visual transformer compression. In: International Conference on Learning Representations (2021)
56. Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al.: Big bird: Transformers for longer sequences. Advances in neural information processing systems **33**, 17283–17297 (2020)
57. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6881–6890 (2021)
58. Zhou, Y., Moosavi-Dezfooli, S.M., Cheung, N.M., Frossard, P.: Adaptive quantization for deep neural network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
59. Zhu, M.H., Gupta, S.: To prune, or not to prune: Exploring the efficacy of pruning for model compression (2018)
60. Zhu, M., Tang, Y., Han, K.: Vision transformer pruning. arXiv preprint arXiv:2104.08500 (2021)
61. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable {detr}: Deformable transformers for end-to-end object detection. In: International Conference on Learning Representations (2021), `https://openreview.net/forum?id=gZ9hCDWe6ke`