SNP: Structured Neuron-level Pruning to Preserve Attention Scores

Kyunghwan Shim[1], Jaewoong Yun[1], and Shinkook Choi[1]*

[1] Nota Inc., Korea

[2] {kyunghwan.shim,jwyun,shinkook.choi}@nota.ai

Abstract. Multi-head self-attention (MSA) is a key component of Vision Transformers (ViTs), which have achieved great success in various vision tasks. However, their high computational cost and memory footprint hinder their deployment on resource-constrained devices. Conventional pruning approaches can only compress and accelerate the MSA module using head pruning, although the head is not an atomic unit. To address this issue, we propose a novel graph-aware neuron-level pruning method, Structured Neuron-level Pruning (SNP). SNP prunes neurons with less informative attention scores and eliminates redundancy among heads. Specifically, it prunes graphically connected query and key layers having the least informative attention scores while preserving the overall attention scores. Value layers, which can be pruned independently, are pruned to eliminate inter-head redundancy. Our proposed method effectively compresses and accelerates Transformer-based models for both edge devices and server processors. For instance, the DeiT-Small with SNP runs 3.1 times faster than the original model and achieves performance that is 21.94% faster and 1.12% higher than the DeiT-Tiny. Additionally, SNP accelerates the efficiently designed Transformer model, EfficientFormer, by 1.74 times on the Jetson Nano with acceptable performance degradation. Source code is at https://github.com/Nota-NetsPresso/SNP

Keywords: Network pruning · Network compression · Compact models

1 Introduction

Vision Transformers (ViTs) [7,17,24] have outperformed or matched the performance of state-ofthe-art convolutional neural networks (CNNs) [10,22,23,28] on various computer vision tasks. The success of ViTs is attributed to the Multi-head Self-Attention (MSA) module, which captures intricate relationships in data. However, the Transformer architecture entails substantial computational resources, posing challenges for practical applications on edge devices with constrained storage and computational capabilities. To address this issue, we leverage the graphical components of the MSA module to reduce the dimension of interconnected layers, aimed at effectively reducing their computing budgets while achieving hardware-agnostic speedups.

* Corresponding author

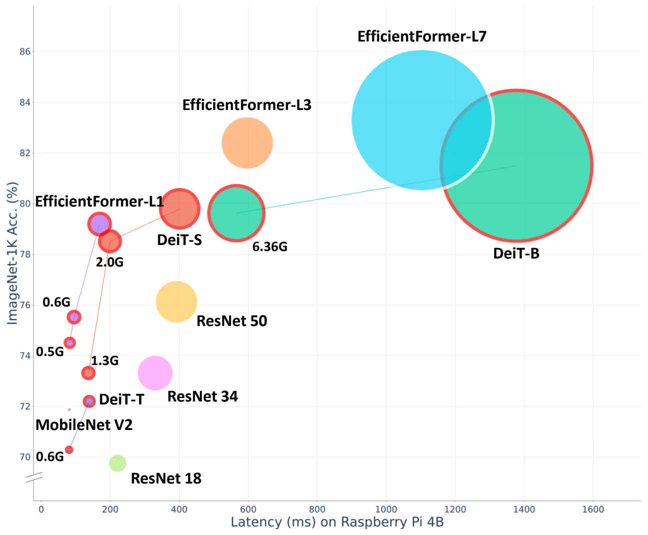

Fig. 1: Comparison of model size, speed, and performance. ImageNet-1K classification results. Latency is profiled by Rasbperry Pi 4B. The connected lines represent the compressed models paired with the original model. The size of each circle indicates the number of parameters in respective model. The number adjacent to each compressed model indicates its compressed FLOPs.

From the perspective of structured pruning, which aims to reduce the number of dimensions in convolutional or linear layers, the MSA module contains two prunable objectives: heads and neurons. Head pruning, which removes the number of heads, is relatively intuitive to implement due to the reduced complexity of graphical elements compared to neuron-level pruning. In contrast, neuron-level pruning reduces the dimension of individual layers in each head of the MSA module, necessitating a comprehensive understanding of the graphical connectivity of the MSA module. A recent work [31] implements neuron-level pruning by zeroing out individual filters without considering the model's graphical connectivity. This indiscriminate zeroing can negatively affect both accuracy and throughput performance [26].

In this paper, we introduce a novel graph-aware neuron-level pruning method called Structured Neuron-level Pruning (SNP) to accelerate and compress ViTs effectively. We propose two pruning criteria based on the function of each layer within the MSA module. SNP prunes filter pairs of query and key layers containing fewer contributions to attention scores. Moreover, SNP aims to reduce redundancy across heads by eliminating the redundant filter from the value layer.

Furthermore, by removing identical filter indices for all the graphically connected layers, SNP could accelerate various Transformer models on various devices without additional libraries, as shown in Fig. 1.

In summary, the major contributions of this paper are:

- We propose a novel graph-aware neuron-level pruning method, SNP, for Transformer models. SNP is the first method to use the graphical characteristics of the MSA module to measure the importance score of neurons.
- To the best of our knowledge, this is the first work to accelerate Transformer models on any device using neuron-level pruning only.
- SNP achieves significant acceleration while maintaining the original performance on several models. Compressed DeiT-Small outperforms DeiT-Tiny by 1.12% in accuracy, with similar FLOPs, and reduces inference time on various edge devices. Furthermore, the proposed method accelerates the efficiently designed Transformer model, EfficientFormer [16], more than two times with acceptable performance degradation.

2 Related Work

2.1 Compressing vision Transformers

The ViTs [7, 17, 24] achieve high performance in numerous vision tasks without specialized image processing modules such as convolutions. The key concept of ViTs is to segment images into patch sequences, convert these patches to token embeddings, and then process them through Transformer encoders [25]. ViTs consist only of Transformer blocks, making them likely to be over-parameterized. For this reason, recent works have aimed to reduce computational cost [3,24] and be memory efficient [18, 26]. DeiT [24] proposes lightweight ViT architectures through knowledge distillation [13]. ToMe [3] proposes accelerating ViTs by directly combining similar tokens, without the need for training. Liu *et al.* [18] propose post-training quantization method using nuclear norm based mixed-precision scheme that does not require fine-tuning for the Vision Transformer.

2.2 Pruning vision Transformers

Unstructured and structured pruning Pruning methods can be broadly categorized into two types, unstructured and structured pruning. Unstructured pruning sets individual weights or parameters to zero, resulting in irregular sparse matrices [9, 14]. Compressed models using unstructured pruning tend to maintain relatively high performance for a given pruning ratio. However, they necessitate additional libraries, such as cuSPARSE [5], Automatic SParsity [20], or SparseDNN [27] to accelerate sparse matrix computations.

Structured pruning, on the other hand, involves the removal of entire groups of units, such as filters or attention heads. This can be implemented using "masking" (zeroing out) [11, 12, 32, 33], or by "pruning" [8, 15]. Structured pruning by masking [11, 12, 32, 33] simply sets the group of units to zero, which requires additional libraries to accelerate the model, as unstructured pruning. "Pruning" [8, 15], on the other hand, requires a comprehensive understanding of the network's graphical connectivity, including element-wise operations that enforce the same input shape. By considering the graphical connectivity and pruning identical filter indices for inter-connected layers, structured pruning can achieve acceleration on any devices.

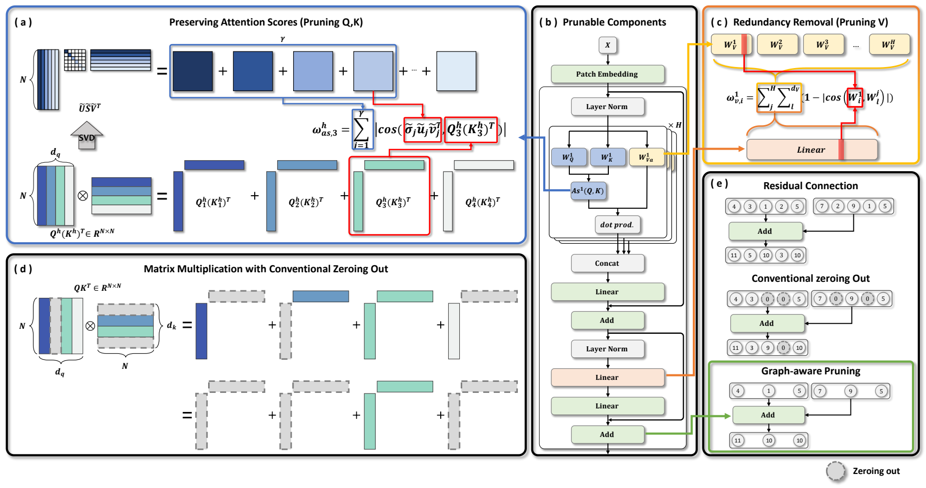

Fig. 2: Proposed SNP methods, on each prunable component of the Transformer block. (a) SNP for query and key layers to preserve attention scores. **(b)** prunable components of the Transformer block. **(c)** SNP for value and other layers, including FFN and patch embedding. **(d)** conventional zeroing out in the matrix multiplication operator. **(e)** conventional zeroing out and graph-aware pruning in the residual connection.

Head and neuron-level pruning Structured pruning for the MSA module has two pruning objectives: head and neuron. Head pruning [29, 30] reduces the number of heads, while neuron-level pruning [31] reduces the dimension of each query, key, and value layer in each head. Recent studies for pruning ViTs have focused on head pruning. X-Pruner [30] proposes a novel head pruning method for ViTs that introduces explainability-aware masks and measure the head importance, resulting in superior model compression. WDPruning [29] propose a method to control the number of attention heads and blocks via threshold on learnable parameters.

UVC [31] utilizes knowledge distillation alongside several pruning techniques, such as head pruning, block pruning, and neuron-level pruning. However, neuron-level pruning of UVC is carried out in a masking (zeroing out) manner, converting the weight matrix into a sparse matrix. For this reason, UVC necessitates additional libraries or hardware for accelerating sparse matrices, otherwise, the compressed model with UVC cannot achieve latency gain from the neuron-level pruning.

3 Methodology

3.1 Preliminaries

MSA module takes a single input $X \in \mathbb{R}^{N \times d}$, where N denotes the input vector length, and d represents the hidden size. It comprises H heads, each consisting of three linear layers: query, key, and value. Each layer denoted as $W^h_{\{q,k,v\}} \in \mathbb{R}^{d \times d_{\{q,k,v\}}}$, where h represents h-th head and $d_{\{q,k,v\}}$ indicates the hidden size of each query, key, and value layer. $\{Q, K, V\}^h$ signifies the output of each query, key, and value layer in h-th head, with a shape of $\mathbb{R}^{N \times d_{\{q,k,v\}}}$.

The self-attention operation for h-th head can be expressed as follows:

$$\mathrm{As}^h(X) = Q^h \cdot (K^h)^T \tag{1}$$

$$\mathrm{Att}^h(X) = \mathrm{Softmax}(\frac{\mathrm{As}^h(X)}{\sqrt{d_q}}) \cdot V^h \tag{2}$$

$$\mathrm{MSA}(X) = \mathrm{Concat}(\mathrm{Att}^1(X), ..., \mathrm{Att}^H(X)) \tag{3}$$

here, As, Att, and MSA represent the functions to calculate attention scores, attention module, and MSA module respectively.

Matrix multiplication, unlike a residual connection, yields zero when either of the inputs is masked, regardless of the value of the other input. Therefore, applying a conventional zeroing-out approach to neuron-level pruning in the MSA module can lead to unintended results, as shown in Fig. 2 (d). To address this issue, we introduce two graph-aware neuron-level pruning criteria to compress and expedite the MSA module.

3.2 Preserving attention scores

Attention scores, Eq. (1), of the MSA module learn long-range dependencies between image features by focusing on different parts of the image when processing different features. These attention scores can be recognized as a series of outer product of Q^h_i and $(K^h_i)^T$ as follows:

$$\begin{aligned}\mathrm{As}^h(X) &= Q^h \cdot (K^h)^T \\ &= \sum_{i=1}^{d_q} Q^h_i \cdot (K^h_i)^T\end{aligned} \tag{4}$$

from this perspective, neuron-level pruning, reducing the dimension of query d_q and key d_k, inevitably distorts the attention scores. For this reason, preserving attention scores is essential to maintain the high performance of the original model.

To alleviate the distortion, we maintain the graphically connected query-key filter pair (Q^h_i and K^h_i), constituting a filter-by-attention score ($Q^h_i \cdot (K^h_i)^T \in \mathbb{R}^{N \times N}$), that retains the most significant aspects of the overall attention scores. To identify the most informative filter pair, we initially employ singular value decomposition (SVD) to decompose the original model's attention scores.

$$\begin{aligned}\mathrm{As}^h(X) &= \tilde{U} \cdot \tilde{S} \cdot \tilde{V}^T \\ &= \sum_{j=1}^{N} \tilde{\sigma}_j \cdot \tilde{u}_j \cdot \tilde{v}^T_j\end{aligned} \tag{5}$$

where $\tilde{U}$ and $\tilde{V}$ are the left and right singular vector matrices, respectively, and $\tilde{S}$ is the diagonal matrix of singular values, with $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq ... \geq \tilde{\sigma}_N$.

SVD is a technique for extracting the most informative components of a matrix, those with large singular values, while discarding the less informative components, those with small singular values. To retain the spatial relationships captured by the attention mechanism, we prune the filter pair Q_i^h and K_i^h with least correlation with the most informative components of the attention scores.

To measure the correlation, we adopt the cosine similarity between the attention scores of the i-th filter and the j-th rank matrix, as normalizing the attention scores reduces the impact of magnitude-based approaches. Consequently, the importance score $\omega_{as,i}^h$ is defined as follows:

$$\begin{aligned} \omega_{as,i}^h &= \sum_{j=1}^{r} |\cos((Q_i^h \cdot (K_i^h)^T), (\tilde{\sigma}_j \cdot \tilde{u}_j \cdot \tilde{v}_j^T))| \\ &= \sum_{j=1}^{r} |\frac{(Q_i^h \cdot (K_i^h)^T) \cdot (\tilde{\sigma}_j \cdot \tilde{u}_j \cdot \tilde{v}_j^T)}{||(Q_i^h \cdot (K_i^h)^T)|| \cdot ||(\tilde{\sigma}_j \cdot \tilde{u}_j \cdot \tilde{v}_j^T)||}| \end{aligned} \tag{6}$$

where r is a hyperparameter dictating the quantity of ranks, within the range of $1 \leq r \leq N$, to be compared with the i-th attention scores, while the remaining $N - r$ singular values $(\tilde{\sigma}_{r+1}, ..., \tilde{\sigma}_N)$ are discarded.

Optimizing r for each attention module can contribute to preserving informative filters and sustaining higher performance. For this reason, we adopt Variational Bayesian Matrix Factorization (VBMF) [21] to determine the optimal rank r for the attention scores. VBMF inherently addresses the challenge of determining the number of latent factors (ranks) in matrix factorization using its probabilistic approach.

Importance score $\omega_{as,i}^h$, defined in Eq. (6), represents the importance of the i-th filter for both query and key layers. A larger $\omega_{as,i}^h$ indicates that the filter has a greater impact on the main component of the attention scores, while a lower $\omega_{as,i}^h$ indicates that the filter is less important and can be removed without significantly affecting the attention scores.

3.3 Inter-head redundancy removal

In the preceding section, we outlined the approach to preserving attention scores even with reduced embedding dimensions in the query and key layers. Here, we introduce a pruning method for the value and other layers, such as FFN or patch embedding layer.

Previous works [2,19] have revealed that a significant proportion of attention heads can be removed without causing significant performance deterioration. To remove this inter-head redundancy through neuron-level pruning, we propose to measure the distance between all the value layers of MSA module, irrespective of the heads. The importance score $\omega_{v,i}^h$ of the i-th filter in the value layer of the

h-th head is as follows:

$$\omega_{v,i}^{h} = \sum_{j=1}^{H} \sum_{l=1}^{d_v} (1 - |\cos(W_i^h, W_l^j)|) \tag{7}$$

The importance score of the value layer, denoted as $\omega_{v,i}^{h}$, indicates the redundancy of filter i in the value layer compared to all value layers of the heads within the corresponding MSA layer. Consequently, filters with the lowest importance scores will be pruned in the initial stages of the pruning process.

3.4 Accelerating Transformers

As described in Sec. 3.2, SNP removes the least correlated filter pairs (Q_i^h, K_i^h) to preserve attention scores and enhance the efficiency of the MSA module across various devices.

Most MSA modules, including MSA modules in DeiTs, are designed to compute all heads in parallel. For achieving this parallel computation, SNP ensures that the number of filter dimensions are consistent across heads, even though the filter indices for each head are independently selected.

The green highlighted boxes in Fig. 2 (b) represent layers connected by a single residual connection at the last add layer of the Transformer block. Unlike the matrix multiplication operator, the output of the residual connection becomes zero when all the interconnected layers return zero for specific filter indices. For this reason, the residual connection with masking always exceeds the performance of actual pruning, which restricts the set of possible pruning patterns [26].

To accelerate the residual connection layers, all interconnected layers should be pruned identically. To achieve this, we sum up all the calculated importance scores for interconnected layers based on their filter indices, as shown in Fig. 2 (e). Subsequently, we prune the least important filter indices of all the connected layers.

4 Experimental Results

To ensure a fair comparison with existing methods, we apply SNP to prune the DeiT [24] architectures trained on the ImageNet-1K [6] dataset. Additionally, we conduct experiments on the efficient Transformer model, EfficientFormer-L1, to confirm the robustness of SNP. Furthermore, a series of ablation studies are conducted to gain a comprehensive understanding of our methodology.

4.1 Implementation details

The overall pruning and fine-tuning process are executed on the pre-trained DeiT[3] and EfficientFormer-L1[4] released from the official implementation on

[3] https://github.com/facebookresearch/deit
[4] https://github.com/snap-research/EfficientFormer

Table 1: Performance comparison of various pruning methods on ImageNet-1K. The “Pruning" column represents the pruning methods that corresponding methods use to compress the MSA module. Each of the methods contains one of follows: head pruning (HP), block pruning (BP), neuron-level pruning (NP), neuron-level sparsity (NS).

Model	Method	Pruning	Top-1 (%)	Top-5 (%)	GFLOPs	Params (M)
DeiT-T	Original [24]	-	72.20	91.10	1.3	5.7
	SSViTE [4]	HP	70.12	-	0.9	4.2
	WDPruning [29]	HP+BP	70.34	89.82	0.7	3.5
	X-Pruner [30]	HP	71.10	90.11	0.6	-
	SNP	**NP**	**70.29**	**90.01**	**0.6**	**3.0**
	UVC [31]	HP+NS+BP	70.60	-	0.5	-
DeiT-S	Original [24]	-	79.85	95.00	4.6	22.1
	SSViTE [4]	HP	79.22	-	3.1	14.6
	WDPruning [29]	HP+BP	78.38	94.05	2.6	13.3
	X-Pruner [30]	HP	78.93	94.24	2.4	-
	UVC [31]	HP+NS+BP	78.82	-	2.3	-
	SNP	**NP**	**78.52**	**94.37**	**2.0**	**10.0**
	SNP	**NP**	**73.32**	**91.66**	**1.3**	**6.4**
DeiT-B	Original [24]	-	81.80	95.59	17.6	86.6
	SSViTE [4]	HP	82.22	-	11.8	56.8
	WDPruning [29]	HP+BP	80.76	95.36	9.9	55.3
	X-Pruner [30]	HP	81.02	95.38	8.5	-
	UVC [31]	HP+NS+BP	80.57	-	8.0	-
	SNP	**NP**	**79.63**	**94.37**	**6.4**	**31.6**

ImageNet-1K. Throughout the fine-tuning phase of the pruned model, we maintain consistent settings across all models, except for the batch size and learning rate. The batch size is set to 256, and to prevent weight explosion, we adjust the learning rate of the compressed model to 1/10 or 1/100 of the original model.

To evaluate the reduced latency using SNP, we have configured four testing scenarios: one on a CPU and another on GPU for both edge devices and server processors. We employ a standard PyTorch model for profiling on the server processors (Intel Xeon Silver 4210R and NVIDIA GeForce RTX 3090). Profiling on the Raspberry Pi 4B and Jetson Nano is conducted using the ONNX and TensorRT formats, respectively. All latencies are measured using a single image as an input, except for the GPU of the server processor (RTX 3090), where it is set to 64 images.

4.2 Quantitative results

Comparison with other methods Despite the constraints outlined in Sec. 3.4, SNP not only maintains accuracy comparable to existing methods but also significantly reduces inference time across diverse hardware and data types on the ImageNet-1K dataset, as shown in Tab. 1 and Tab. 2.

Table 2: Inference speed and Top-1 accuracy of the compressed model across different devices. Performance evaluation involves accuracy on ImageNet-1K and inference time for the compressed DeiTs and EfficientFormer-L1. Latency is benchmarked with 200 warm-up runs and averaged over 1000 runs. In latency measurement, a single image is used as an input, except for the RTX 3090, where 64 images are employed in a single batch.

Model	Top-1 (%)	GFLOPs	Edge devices (ms)		Server processors (ms)	
			Raspberry Pi 4B (.onnx)	Jetson Nano (.trt)	Xeon Silver 4210R (.pt)	RTX 3090 (.pt)
DeiT-Tiny	72.20	1.3	139.13	41.03	34.74	18.65
+ **SNP (Ours)**	**70.29**	**0.6**	**81.63 (1.70×)**	**26.67 (1.54×)**	**25.25 (1.38×)**	**17.82 (1.05×)**
DeiT-Small	79.80	4.6	401.27	99.32	53.37	46.13
+ **SNP (Ours)**	**78.52**	**2.0**	**199.15 (2.01×)**	**45.51 (2.18×)**	**38.57 (1.38×)**	**32.91 (1.40×)**
+ **SNP (Ours)**	**73.32**	**1.3**	**136.68 (2.94×)**	**32.03 (3.10×)**	**33.46 (1.60×)**	**26.98 (1.71×)**
DeiT-Base	81.80	17.6	1377.71	293.29	122.03	151.35
+ **SNP (Ours)**	**79.63**	**6.4**	**565.68 (2.44×)**	**132.55 (2.21×)**	**64.65 (1.89×)**	**72.96 (2.07×)**
EfficientFormer-L1	79.20	1.3	169.13	30.95	43.75	26.19
+ **SNP (Ours)**	**75.53**	**0.6**	**95.12 (1.78×)**	**19.78 (1.56×)**	**38.25 (1.14×)**	**17.24 (1.52×)**
+ **SNP (Ours)**	**74.51**	**0.5**	**82.60 (2.05×)**	**17.76 (1.74×)**	**35.15 (1.24×)**	**16.01 (1.64×)**

In a recent study [26], unconstrained masking generally outperforms post-training accuracy of pruned models by an average of 2.1% on ImageNet-1K. Compared to unconstrained head masking approaches like SSVITE [4] and X-Pruner [30], SNP achieves significantly higher compression rates for all DeiTs FLOPs (30.64% and 10.64%) with minimal performance degradation (0.83% and 0.87%), much less than the 2.1% average mentioned.

Furthermore, compared to other pruning approaches like WDPruning [29] and UVC [31], which use a combination of pruning techniques to compress DeiTs, SNP achieves comparable accuracy solely through neuron-level pruning. Notably, DeiT-Small with SNP outperforms WDPruning by 0.14%, with the removal of 3.3 million parameters and a reduction of 0.6 GFLOPs. Compared to UVC, SNP exhibits negligible performance degradation, averaging 0.51% across all DeiTs while using 7.65% fewer FLOPs.

Large compressed vs. Small hand-crafted DeiT-Small with SNP, a large pruned model, outperforms the smaller, hand-crafted DeiT-Tiny in both accuracy and latency, achieving a notable 1.12% improvement in top-1 accuracy while maintaining similar FLOPs. Additionally, DeiT-Small with SNP exhibits enhanced speed compared to the original DeiT-Tiny across edge devices and CPU-based server processors. Notably, its speed increases up to 21.94% compared to the original DeiT-Tiny running on Jetson Nano, a GPU-based edge device. This substantial performance gap underscores the superiority of the compressed model (DeiT-Small with SNP) over the smaller hand-crafted designed model (DeiT-Tiny) in both overall performance and speed.

Accelerating Transformer-based models As depicted in Tab. 2, SNP achieves impressive acceleration of DeiTs by a factor of 1.44× to 2.44× on edge devices and 1.05× to 2.07× on server processors. This acceleration is notable,

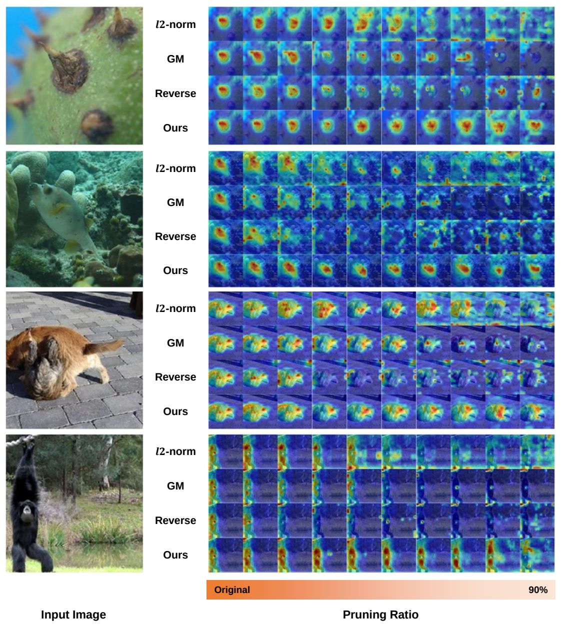

Fig. 3: Attention maps with varying pruning criteria and compression ratios. All query and key layers are locally pruned based on the specified pruning ratio without fine-tuning. The importance scores of $l2$-norm and GM on query and key layers are combined by filter index and pruned simultaneously. “Reverse" represents the reverse order of SNP.

with negligible average performance degradation of 1.79%, specifically 1.91%, 1.28%, and 2.17% for DeiT-Tiny, DeiT-Small, and DeiT-Base, respectively.

Compared to WDPruning, which employs both head and block pruning, SNP surpasses in terms of latency for DeiT-Small and DeiT-Base on the RTX 3090 GPU. SNP accelerates the original DeiTs by 1.38× and 2.07×, respectively, while WDPruning achieves a comparatively modest acceleration of 1.18× for both models.

The superiority of SNP becomes more evident when the target of neuron-level pruning, linear or convolutional layers, contribute a larger proportion of the original model's computation time. we find that SNP further accelerates original model 1.05× to 2.07× as the DeiT model's size grows on the RTX 3090 GPU. In contrast, WDPruning, involving the removal of entire layers and associated operators in both head and block, consistently shows acceleration rates of 1.18×.

To ascertain the robustness of SNP across diverse Transformer models, especially on the efficiently designed model, we conduct additional experiments on EfficientFormer-L1. SNP accelerates the model by 1.78× and 2.05× faster on Raspberry Pi 4B and 1.56× and 1.74× faster on Jetson Nano. Notably, the compressed EfficientFormer-L1 achieves acceptable accuracy of 75.53% and 74.51%.

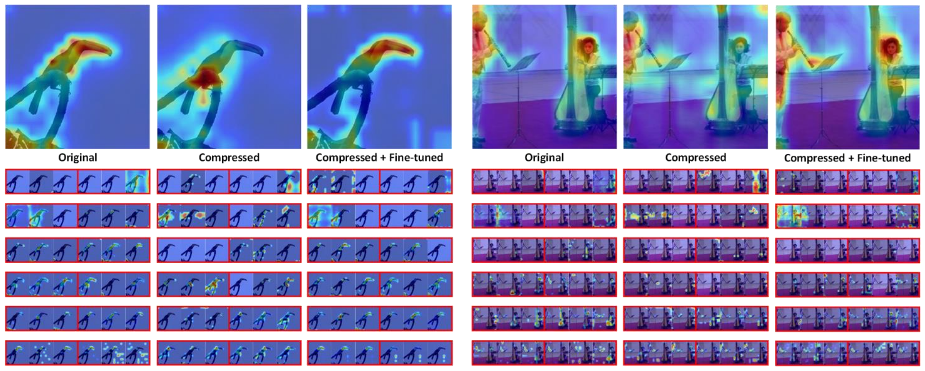

Fig. 4: Attention maps from the original, compressed, and fine-tuned DeiT-Tiny with SNP. The attention maps in the first row are visualized using the attention rollout [1]. Each red box contains three attention maps from each head of the MSA module, ordered accordingly.

4.3 Qualitative results

In the subsequent sections, attention rollout [1] is employed to randomly selected images from the test datasets to visualize the attention maps of the DeiTs and assess the efficacy of SNP. This evaluation unfolds across two key facets:

– **Preserving the attention scores**: Visualize attention maps to verify the effectiveness of the proposed importance score in preserving attention scores.
– **Restoring the attention scores after SNP**: Visualize attention maps to find the effectiveness of SNP in restoring attention scores.

Preserving the attention scores To visualize SNP effectiveness, we apply four pruning criteria to compress the query and key filter pairs in all MSA modules of DeiT-Tiny: SNP, $l2$-norm, geometric median (GM) [11], and “Reverse".

The “Reverse" criterion prioritizes pruning the most important filter pairs first, keeping the least important pairs until the end. However, both $l2$-norm and GM pruning criteria ignore the MSA module's graphical connectivity, evaluating importance scores independently for query and key layers. To handle identical filter indices during pruning, we aggregate scores based on filter indices, removing the least important indices from both layers.

In Fig. 3, attention maps for the original DeiT-Tiny and locally pruned models are presented across various pruning ratios (10% to 90% with 10% intervals). Our proposed method effectively maintains the original attention map even after pruning over 80% of the filters, whereas other methods show fragmented attention maps at much lower pruning ratios (typically 30% or less). These results highlight the potential of our neuron-level pruning criteria, utilizing SVD to preserve attention scores, in reducing the size and speeding up the execution of MSA modules without compromising accuracy.

Table 3: Performance of SNP without fine-tuning across different data quantities at various pruning ratios. To determine the proper number of images for calculating the importance score, we conduct local pruning on the query and key layers at pruning ratios of 10%, 30%, 50%, 70%, and 90%. All performance metrics are assessed without fine-tuning. The latency is measured on Raspberry Pi 4B.

Number of images	**Performance by pruning ratio**					
	Original	**10%**	**30%**	**50%**	**70%**	**90%**
1	72.20	71.15	67.96	57.60	38.68	11.17
4	72.20	70.82	67.86	57.65	38.05	11.86
16	72.20	70.72	67.42	58.60	39.46	9.11
64	72.20	70.83	67.55	59.50	42.13	12.83
256	72.20	70.75	68.14	59.58	42.70	14.48
Latency (ms)	139.13	133.60	129.29	119.41	117.48	112.32
Params (M)	5.7	5.59	5.41	5.23	5.06	4.88

Restoring the attention scores after SNP Fig. 4 illustrates the overall and per-head attention maps of the original, compressed, and fine-tuned DeiT-Tiny, respectively. The first row shows the overall attention maps of the respective models. Notably, the compressed model maintains a well-preserved overall attention map, despite pruning all layers, including values and FFN, resulting in a 53% reduction in FLOPs and a 46% reduction in parameters. Especially, we can observe that attention map is well-restored after the fine-tuning process.

The twelve red boxes below the overall attention map depict per-head attention maps for twelve layers of each original, compressed, and fine-tuned models respectively. As shown in Fig. 4, it is evident that the attention maps for each head are effectively preserved and restored in each of the compressed and fine-tuned models.

4.4 Ablation studies

Importance scores by the data quantity Since the attention scores depend on the input, as indicated in Eq. (1), the proposed importance scores for query and key filter pairs (Eq. (6)) may exhibit sensitivity to the distribution of the input image X. To validate the method's robustness, we compute importance scores using various image quantities, pruning query and key layers at different ratios, without fine-tuning process.

As depicted in Tab. 3, SNP demonstrates a slight advantage in preserving performance with an increasing number of images, outperforming models compressed with fewer images as the compression ratio rises. However, this improvement comes at the cost of increased computation time for SNP calculations. Considering these factors into account, we opt to use 64 images, which yield the second-best performance among the given number of images. This decision strikes a balance between achieving satisfactory performance and maintaining computational efficiency.

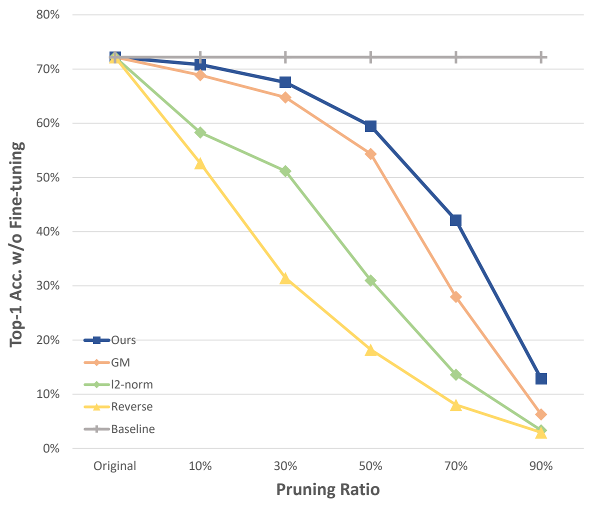

Fig. 5: Top-1 accuracy of compressed DeiT-Tiny on ImageNet using several pruning criteria without fine-tuning. Query and key layers are locally pruned using various pruning criteria : SNP, GM, $l2$-norm, reverse order of SNP ("Reverse"), and original DeiT-Tiny ("Baseline").

Performance comparison across pruning ratios To assess the robustness of SNP, we examine the performance of DeiT-Tiny under various pruning criteria and different pruning ratios, as described in Sec. 4.3.

As illustrated in Fig. 5, SNP consistently outperforms other pruning criteria across all pruning ratios. In contrast, models compressed with "Reverse" criteria exhibit the lowest performance at all pruning ratios, underscoring the robustness of the proposed approach.

Both Fig. 3 and Fig. 5 confirm that SNP successfully preserves attention scores in both quantitative and qualitative aspects. Even with an 80% pruning ratio and without fine-tuning, SNP maintains original attention scores and outperforms other pruning criteria.

5 Conclusion

In this paper, we propose a novel graph-aware neuron-level pruning method, SNP, designed to compress and accelerate Transformer-based models. SNP proposes two pruning criteria for preserving attention scores and eliminating inter-head redundancy. Using SNP, a large compressed model outperforms small, hand-crafted designed models in both performance and latency on edge devices. Moreover, the compressed models exhibit astonishing results in latency on various devices, with negligible performance degradation.

As this work is a first attempt to accelerate MSA modules using neuron-level pruning alone, many challenges remain. One is incorporating other pruning methods such as head or block pruning for a more efficient Transformer model. Another challenge is applying SNP to other vision tasks, including image generation, which requires high computational costs on both training and inference.

We believe that these works encourage the adoption of model pruning as a tool, for both improving the applicability of ViTs in resource-constrained environments and reducing the training costs of large models by integrating with training process.

Acknowledgements

This work was supported by Artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT(MSIT, Korea)&Gwangju Metropolitan City.

References

1. Abnar, S., Zuidema, W.: Quantifying attention flow in transformers. arXiv preprint arXiv:2005.00928 (2020)
2. Bian, Y., Huang, J., Cai, X., Yuan, J., Church, K.: On attention redundancy: A comprehensive study. In: Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: human language technologies. pp. 930–945 (2021)
3. Bolya, D., Fu, C.Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461 (2022)
4. 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)
5. Demouth, J.: Sparse matrix-matrix multiplication on the gpu. Tech. rep., NVIDIA (2012)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
7. 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. arXiv preprint arXiv:2010.11929 (2020)
8. Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X.: Depgraph: Towards any structural pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16091–16101 (2023)
9. 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)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4340–4349 (2019)

12. 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)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
14. Lee, J., Park, S., Mo, S., Ahn, S., Shin, J.: Layer-adaptive sparsity for the magnitude-based pruning. arXiv preprint arXiv:2010.07611 (2020)
15. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
16. Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., Ren, J.: Efficientformer: Vision transformers at mobilenet speed. Advances in Neural Information Processing Systems **35**, 12934–12949 (2022)
17. 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)
18. Liu, Z., Wang, Y., Han, K., Zhang, W., Ma, S., Gao, W.: Post-training quantization for vision transformer. Advances in Neural Information Processing Systems **34**, 28092–28103 (2021)
19. Michel, P., Levy, O., Neubig, G.: Are sixteen heads really better than one? Advances in neural information processing systems **32** (2019)
20. Mishra, A., Latorre, J.A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., Yu, C., Micikevicius, P.: Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378 (2021)
21. Nakajima, S., Sugiyama, M., Babacan, S.D., Tomioka, R.: Global analytic solution of fully-observed variational bayesian matrix factorization. The Journal of Machine Learning Research **14**(1), 1–37 (2013)
22. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10428–10436 (2020)
23. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
24. 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)
25. 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)
26. Wan, A., Hao, H., Patnaik, K., Xu, Y., Hadad, O., Güera, D., Ren, Z., Shan, Q.: Upscale: unconstrained channel pruning. In: International Conference on Machine Learning. pp. 35384–35412. PMLR (2023)
27. Wang, Z.: Sparsednn: Fast sparse deep learning inference on cpus. arXiv preprint arXiv:2101.07948 (2021)
28. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10687–10698 (2020)
29. Yu, F., Huang, K., Wang, M., Cheng, Y., Chu, W., Cui, L.: Width & depth pruning for vision transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 3143–3151 (2022)

30. 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)
31. Yu, S., Chen, T., Shen, J., Yuan, H., Tan, J., Yang, S., Liu, J., Wang, Z.: Unified visual transformer compression. arXiv preprint arXiv:2203.08243 (2022)
32. Yu, S., Mazaheri, A., Jannesari, A.: Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In: International conference on machine learning. pp. 25656–25667. PMLR (2022)
33. Yu, X., Serra, T., Ramalingam, S., Zhe, S.: The combinatorial brain surgeon: pruning weights that cancel one another in neural networks. In: International Conference on Machine Learning. pp. 25668–25683. PMLR (2022)