Sparse Structure Exploration and Re-optimization for Vision Transformer

Sangho An[1] Jinwoo Kim[1] Keonho Lee[2] Jingang Huh[2] Chanwoong Kwak[2] Yujin Lee[2] Moonsub Jin[2]
Jangho Kim*[1]

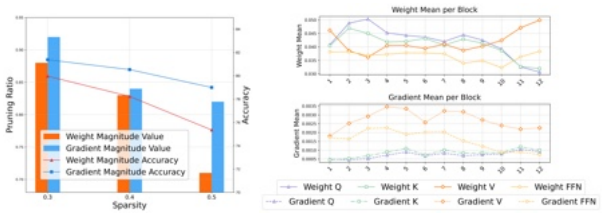
[1] Kookmin University, Seoul, Korea
[2] RoboticsLab , Hyundai Motor Company

Abstract

Vision Transformers (ViTs) achieve outstanding performance by effectively capturing long-range dependencies between image patches (tokens). However, the high computational cost and memory requirements of ViTs present challenges for model compression and deployment on edge devices. In this study, we introduce a new framework, Sparse Structure Exploration and Re-optimization (SERo), specifically designed to maximize pruning efficiency in ViTs. Our approach focuses on (1) hardware-friendly pruning that fully compresses pruned parameters instead of zeroing them out, (2) separating the exploration and re-optimization phases in order to find the optimal structure among various possible sparse structures, and (3) using a simple gradient magnitude-based criterion for pruning a pre-trained model. SERo iteratively refines pruning masks to identify optimal sparse structures and then re-optimizes the pruned structure, reducing computational costs while maintaining model performance. Experimental results indicate that SERo surpasses existing pruning methods across various ViT models in both performance and computational efficiency. For example, SERo achieves a 69% reduction in computational cost and a 2.4x increase in processing speed for DeiT-Base model, with only a 1.55% drop in accuracy. Implementation code: https://github.com/Ahnho/SERo.

1 INTRODUCTION

Vision Transformers (ViTs), in particular, have shown success in numerous computer vision tasks by tokenizing image patches and utilizing them in various innovative ways [Liu et al., 2021, Touvron et al., 2021]. Despite their successes, however, the excessive parameter count and computational cost of transformers have become significant barriers to the efficiency of ViTs. Consequently, extensive research has focused on pruning parameters to enhance the computational efficiency of ViTs.



(a) Value Ratio and Accuracy (b) Weight and Gradient distributions

Figure 1: Figure 1a shows the accuracy and remaining value ratio after weight and gradient magnitude pruning according to sparsity ratio on CIFAR-100 dataset of DeiT-Tiny model, and Figure 1b illustrates the weight and gradient distributions of Q, K, V, and FFN layers in the pre-trained model with 128 batch size.

According to [Paul et al., 2023], well explored sparse structure of networks often exhibit a flatter error landscape, which provides a favorable structure for training and thus offers optimization advantages. Notably, dynamic pruning methods [Lin et al., 2020c, Kim et al., 2023] emphasize exploring various sparse structures during training can lead to finding better sparse structures. Due to these benefits, the importance of exploration in sparse structures becomes more pronounced. [Zimmer et al., 2023] highlights that re-optimizing models obtained through one-shot pruning can lead to better-performing sparse models. Considering previous research, finding an optimal sparse structure is essential. A well-explored structure enables re-optimization to begin from a more advantageous landscape, and instead of merely fine-tuning the sparse model, re-optimizing it is crucial for recovering any lost performance.

*Corresponding Author

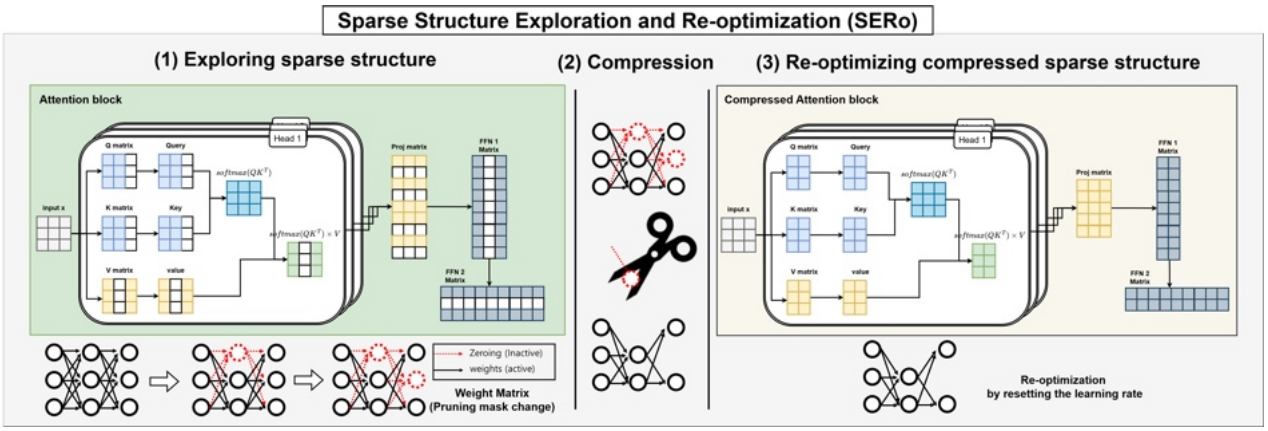


Figure 2: The overall process of our proposed SERo method consists of three main steps. **First**, we explore the sparse structure by dynamically adjusting masks while considering the dimensions of multi-head self-attention and the loss of the value matrix. **Second**, we compress the discovered structure to achieve actual computational speed improvements. **Finally**, we reset the learning rate and re-optimize with the identified structure with few epochs.

However, In the case of ViT pruning, there are many considerations due to the need to fine-tune a pre-trained model with an attention mechanism for downstream tasks. Although existing methods provide effective solutions, they still have limitations. For example, if pruning is performed without considering dimensionality and simply zeros out inactive channels, it may fall short in achieving practical gains in the computational cost and the latency [Yu et al., 2022b,c]. When selecting channels (neurons) to prune, criteria with high computational costs, such as importance or Hessian, are often used, and the model is fixed in the early epochs rather than focusing on exploring various sparse structures [Shim et al., 2024, Chen et al., 2021]. Additionally, the identified sparse structure is learned through fine-tuning rather than re-optimization [Yu et al., 2022a, Yu and Xiang, 2023, Yu et al., 2022d, He et al., 2017, 2019].

As shown in Figure 1, we observe that the weights in the Q and K matrices of the ViT's pre-trained model generally have higher magnitudes compared to those in the V matrix. Consequently, when applying a typical weight magnitude-based pruning approach [Han et al., 2015, Yu et al., 2022b] to derive sparse structures, the value matrix tends to be pruned more extensively, leading to a proportional decline in performance. In the attention mechanism, even if Q and K matrices are heavily pruned, information is still preserved across tokens. However, significant pruning of the value matrix results in considerable information loss. This effect becomes more pronounced as the pruning ratio increases, demonstrating that even a simple criterion based on gradient magnitude can effectively guide channel selection without requiring complex criteria.

Based on our observation, to address these challenges while leveraging the strengths of structured exploration and re-optimization suited for ViT, this work introduces a novel ViT pruning methodology from three perspectives:

(1) Real compression for computational costs and latency efficiency: We fully compress the pruned parameters instead of merely zeroing them out, leading to tangible improvements in both computational costs and the inference speed. **(2)** Separation of exploration and re-optimization: Unlike previous methods, we separate the exploration phase to find an optimal sparse model from the re-optimization phase. After identifying an appropriate sparse network structure, we compress and perform re-optimization by resetting learning rate instead of fine-tuning. **(3)** Simple criterion considering transfer learning: By analyzing the characteristics of pre-trained models, we propose a simple criterion for effectively exploring sparse models in the context of transfer learning.

We propose a gradient-based sparse structure exploration and re-optimization (SERo) that incorporates three perspectives. Figure 2 shows the overall process of SERo. To select an optimal sparse model based on a pre-trained model, we employ a simple gradient magnitude criterion. For broader exploration, we use gradual pruning with dynamic pruning that periodically updates the pruning mask. Since weight magnitude is known to significantly impact model performance in most pruning methods, we design our approach to prioritize updates to parameters with higher weight magnitudes among those identified as important by the gradient criterion. This enables effective exploration and discovery of high-quality sparse structures. Once the target sparsity is reached, exploration is halted, and compression is applied to establish the sparse structure. Unlike typical fine-tuning, we separate exploration from re-optimization, allowing us to re-optimize from a well-structured sparse foundation and thereby maximize the performance of the sparse model. Our contributions are as follows:

- We introduce a novel sparse structure exploration and re-optimization method (SERo) based on three perspectives, incorporating both gradient and weight

magnitude in the exploration phase and applying re-optimization on the compressed sparse structure.

- We provide a theoretical analysis of the convergence properties of our exploration update method and investigate the advantages of re-optimizing the explored sparse structure, analyzing differences from other exploration techniques.
- We demonstrate the effectiveness across various benchmarks and validate the effectiveness of our approach in the latency and accuracy through extensive experiments. In particular, DeiT-Base with SERo improve 2.4x faster than the original model with minimal performance drop.

The proposed method maintains ViT performance while achieving practical computational costs and speed optimizations suitable for deployment. This new approach to ViT pruning demonstrates the potential to maximize model efficiency and transfer learning performance, making it highly applicable to real-world scenarios.

2 RELATED WORK

2.1 PRUNING

Unstructured pruning [Lin et al., 2020c, Frankle and Carbin, 2019, Liu et al., 2019] removes less important weights at the individual level. While this approach can achieve higher compression rates, it is difficult to accelerate with hardware. On the other hand, structured pruning [Luo et al., 2017, He et al., 2018, Lin et al., 2020b,a] removes larger structural units, such as channels, filters, or layers, effectively altering the network's architecture. As a result, it can directly improve hardware acceleration. Our proposed SERo adopts this structured pruning approach to effectively achieve performance improvements on actual hardware.

2.2 VISION TRANSFORMER MODEL PRUNING

Vision Transformers (ViTs)[Dosovitskiy et al., 2021, Liu et al., 2021, Touvron et al., 2021] are models that convert image patches into tokens, embed them, and utilize only the encoder structure of transformers[Vaswani et al., 2023]. While ViTs demonstrate exceptional performance in image processing tasks, they face significant limitations due to their high computational complexity and large memory requirements. Various research efforts are underway to address these challenges by reducing computational costs. One approach involves proposing lightweight ViT architectures through Knowledge Distillation techniques, as demonstrated in DeiT [Touvron et al., 2021]. For model pruning approaches, SSViTE [Chen et al., 2021] proposes a method to optimize model parameters and explore connectivity during training, dynamically extracting and learning sparse subnetworks of ViT. WDPruning [Yu et al., 2022a] introduces compression by simultaneously reducing both the width and depth dimensions using trainable parameters and shallow classifiers. X-pruner [Yu and Xiang, 2023] proposes an end-to-end explainability-aware mask to measure each prunable unit's contribution. UVC [Yu et al., 2022b] presents an integrated framework combining pruning, layer skipping, and knowledge distillation. More recently, SNP [Shim et al., 2025] introduces structured neuron-level pruning, which prunes the queries and keys with the least information while maintaining the overall attention scores. All these approaches share the common goal of improving ViT efficiency while maintaining or enhancing their performance.

3 PROPOSED METHOD

In this section, we describe the basic structure of the ViT and explain the pruning granularity, criterion, and the exploration of the granular sparse structure used in the proposed method and the Re-optimization.

3.1 PRELIMINARY

A ViT block consists of a self-attention layer, projection layer and the feed-forward network in the form of a multi-layer perceptron. The self-attention layer transforms the input tokens $X \in \mathbb{R}^{N \times E}$ into query, key, and value representations using weight matrices $W^q, W^k, W^v \in \mathbb{R}^{E \times E}$, resulting in $Q, K, V \in \mathbb{R}^{N \times E}$. To capture a diverse set of attentions, multi-head self-attention (MSA) is applied by splitting Q, K, V into H heads. Each head performs a self-attention operation defined as $\text{Attn}^l(Q_l, K_l, V_l) = \text{softmax}\left(\frac{Q_l K_l^T}{\sqrt{E/H}}\right) V_l$, where l is the index of the head, and there are H heads in total. Here, $Q_l, K_l, V_l \in \mathbb{R}^{N \times E/H}$. The resulting attention outputs Attn^l from each head are concatenated and restored to the original dimension $\mathbb{R}^{N \times E}$ before being passed through the feed-forward network. The MSA operation is expressed as follows:

$$\text{MSA}(X) = \text{concat}(\text{Attn}^1, \text{Attn}^2, \ldots, \text{Attn}^H).$$

The values obtained through the diverse attentions of MSA are concatenated, passed through a projection layer W^P, followed by layer normalization, and then processed through Feed Forward Network (FFN) with weights W^{f1} and W^{f2}.

3.2 PRUNING UNIT, GRANULARITY AND CRITERION

To perform structured pruning in a hardware-friendly manner, we divide the pruning units into three groups based on their functional components (Q&K, V&proj, FFN). Therefore, we share the pruning mask M at the level where ac-

tual computations occur. W^q and W^k are pruned using $M^{q,k}$,W^v and W^p are pruned using M^v , and finally, W^{f1} and W^{f2} are pruned using M^f .

To prune W^q and W^k by columns based on the average L_1 norm of their gradients, we define a column-wise pruning mask $M \in \mathbb{R}^E$, applied uniformly across all elements in each column of W^q and W^k. This results in:

$$\overline{W}^q = M^{q,k} \odot W^q, \quad \overline{W}^k = M^{q,k} \odot W^k$$

where $M^{q,k} = \{m^{q,k}i,j \mid m^{q,k}i,j \in \{0,1\}, i = 1,\ldots,E, j = 1,\ldots,E\}$, each entry $M_j^{q,k}$ in the mask vector $M^{q,k}$ is determined by the average L_1 norm of the gradients for the j-th column of W^q and W^k. Specifically, the gradient of the loss $\mathcal{L}$ is computed with respect to the full weight matrices W^q and W^k, and we then select the j-th column of the resulting gradient matrices:

$$M_j^{q,k} = \begin{cases} 1 & \text{if } \frac{1}{2}\left(\|\nabla_{W^q}\mathcal{L}[:,j]\|_1 + \|\nabla_{W^k}\mathcal{L}[:,j]\|_1\right) \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\nabla_{W^q}\mathcal{L}[:,j]$ and $\nabla_{W^k}\mathcal{L}[:,j]$ denote the gradients of the loss $\mathcal{L}$ with respect to the full weight matrices W^q and W^k, indexed by the j-th column. The threshold τ is used to decide which columns are retained or pruned, with columns below this threshold pruned by setting $M_j^{q,k} = 0$. Each entry $M_j^{q,k}$ applies uniformly across all rows in the j-th column of W^q and W^k, such that the entire column is either pruned or retained.

To prune W^v, we use the L_1 norm of the gradient of W^v, and we prune by removing columns, reducing its dimension. To account for the deleted dimensions in the next layer, we also apply pruning to the weights in W^p that correspond to the pruned input dimensions. This results in:

$$\overline{W}^v = M^v \odot W^v, \quad \overline{W}^p = (M^v)^T \odot W^p$$

where $(M^v)^T$ is the transpose operation. The pruning mask M_j^v is determined as follows:

$$M_j^v = \begin{cases} 1 & \text{if } \|\nabla_{W^v}\mathcal{L}[:,j]\|_1 \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Similarly to W^v, we perform pruning in the feed-forward network based on W^{f1}, and the corresponding dimensions are removed in W^{f2}. This results in:

$$\overline{W}^{f1} = M^f \odot W^{f1}, \quad \overline{W}^{f2} = (M^f)^T \odot W^{f2}$$

where the pruning mask M_j^f is determined as follows:

$$M_j^f = \begin{cases} 1 & \text{if } \|\nabla_{W^{f1}}\mathcal{L}[:,j]\|_1 \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3.3 STRUCTURE EXPLORATION AND RE-OPTIMIZATION

The dynamic sparse structure update rule with t-th iteration as follows:

$$W_{t+1} = W_t - \eta \cdot \nabla_{\overline{W}_t}\mathcal{L} \tag{4}$$

has the advantage of allowing dynamic exploration of sparse structures through the Straight Through Estimator (STE) [Lin et al., 2020c, Guo et al., 2016]. However, STE introduces instability because it estimates the non-differentiable part $\nabla_{W_t}\overline{W}_t$ as 1 during the update process [Kim et al., 2023, Zhou et al., 2021]. To ensure more stable sparse structure exploration, we use a soft pruning mask such that $\nabla_{W_t}\overline{W_t} = M_t$, leading to the update rule:

$$W_{t+1} = W_t - \eta \cdot \nabla_{\overline{W_t}}\mathcal{L} \cdot M_t \cdot W^{|\cdot|} \tag{5}$$

We denote $|\cdot|$ as a element-wise absolute function. Note that W consists of $W^q, W^k, W^v, W^p, W^{f1}, W^{f2}$, which are the weight matrices included in the ViT block as described in Section 3.1, and M corresponds to the masks applied to each weight matrix as explained in Section 3.2. In Figure 1, we addressed the issue of imbalance in W of the pre-trained model. To resolve this issue, we conduct a gradient-based pruning. Since the magnitude of W, which reflects the contribution to the computation, was not considered during the gradient-based pruning, we apply the non-uniform scaling of $W^{|\cdot|}$ to the gradient. This approach reflects the intention of updating more heavily for weights with larger magnitudes, as they are more important. Gradual pruning is employed to progressively increase sparsity, influencing the selection of filters to be removed in future iterations of sparse structure discovery [Lin et al., 2020c].

Convergence analysis : We provide a convergence analysis for the proposed dynamic pruning method with weight magnitude alignment in convex functions. The update rule for the weights is defined as:

$$W_{t+1} = W_t - \eta \cdot \nabla_{\overline{W_t}}\mathcal{L} \cdot M_t \cdot W^{|\cdot|}$$

where M is the pruning mask, $W^{|\cdot|}$ represents the element-wise magnitude of the weights, and $\nabla_{\overline{W_t}}\mathcal{L}$ is the gradient with respect to the pruned weights.

Theorem 1. *Let W_* be the optimal solution for a convex loss function $\mathcal{L}(W)$, and assume the gradient of the pruned model is bounded by G^2 for every pruned model, such that: $\mathbb{E}\left[\|\nabla_{\overline{W_t}}\mathcal{L}\|^2 \mid W_t\right] \leq G^2$ and the loss difference between $\mathcal{L}(\overline{W_t}) - \mathcal{L}(W_t) \leq \epsilon$ at every t-th iteration. Then, the convergence of the pruned model with weight magnitude alignment is upper-bounded as follows:*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\overline{W}_t)\right] - \mathcal{L}(W_*) \leq \frac{\|W_1 - W_*\|^2}{2\eta T} + 2\eta T\epsilon + \frac{\eta G^2}{2}\mathbb{E}\left[\|M_t \cdot W_t^{|\cdot|}\|^2\right] \tag{6}$$

Proof. The update rule for the pruned weight is:

$$W_{t+1} = W_t - \eta \cdot \nabla_{\overline{W_t}} \mathcal{L} \cdot M \cdot W^{|\cdot|}$$

Let W_t be fixed, and take the expectation over W_t. Using the convexity of $\mathcal{L}$, we can express the difference between the current weights and the optimal solution as:

$$\begin{aligned}\mathbb{E}\left[\|W_{t+1} - W_*\|^2 \mid W_t\right] = \|W_t - W_*\|^2 + \eta^2 \mathbb{E}\left[\|\nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|}\|^2\right] \\ -2\eta \mathbb{E}\left[\langle \nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|}, W_t - W_* \rangle \mid W_t\right].\end{aligned} \tag{7}$$

By bounding the gradient term and using the assumption that $\mathbb{E}\left[\|\nabla_{\overline{W_t}} \mathcal{L}\|^2 \mid W_t\right] \leq G^2$, we obtain:

$$\begin{aligned}\mathbb{E}\left[\|W_{t+1} - W_*\|^2 \mid W_t\right] \leq \|W_t - W_*\|^2 + \eta^2 G^2 \mathbb{E}\left[\|M_t \cdot W_t^{|\cdot|}\|^2\right] \\ - 2\eta(\mathcal{L}(\overline{W_t}) - \mathcal{L}(W_*) - \epsilon).\end{aligned} \tag{8}$$

Summing over $t = 1$ to T, we arrive at the final result:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\overline{W}_t)\right] - \mathcal{L}(W_*) \leq \frac{\|W_1 - W_*\|^2}{2\eta T} + 2\eta T\epsilon + \frac{\eta G^2}{2} \mathbb{E}\left[\|M_t \cdot W_t^{|\cdot|}\|^2\right]. \tag{9}$$

Details are in Appendix. □

Once the exploration of the sparse structure is complete, we modify the structure of W using the mask M. Since pruning was performed at the column level, making it hardware-friendly, we can remove the parts corresponding to M_j from the matrix through structured pruning. We define the newly generated lightweight network, after the exploration is complete, as $\tilde{W}$. Then, we optimize $\tilde{W}$ with the conventional gradient descent as depicted in Figure 2.

$$\tilde{W}_{t+1} = \tilde{W}_t - \eta \cdot \nabla_{\tilde{W}_t} \mathcal{L} \tag{10}$$

By initializing $\tilde{W}_1$ with the parameters found in the exploration phase ($\tilde{W}_1 := \overline{W}_T$, also represented as $W_1^{\overline{W}}$), the convergence bound is tighter compared to starting with randomly initialized parameters of the same structure.

$$\Delta = \frac{\|W_1^{\text{random}} - W^*\|^2 - \|W_1^{\overline{W}} - W^*\|^2}{2\eta T} \tag{11}$$

4 EXPERIMENT

We applied our proposed method to perform pruning and compression on DeiT [Touvron et al., 2021] models (Tiny, Small and Base), which are pretrained models trained on the ImageNet-1K [Deng et al., 2009] and CIFAR [Krizhevsky et al., 2009] dataset. We followed the same setting of SNP Shim et al. [2025]. We did not add additional epochs for the re-optimization phase; instead, we split the total epochs into the exploration and re-optimization phases, ensuring the same training cost as SNP. Implementation details are provided in Appendix and the implemented code.

Algorithm 1 Sparse Structure Exploration and Re-optimization (SERo)

Require: Total epochs $E = E_e + E_o$, Pruning frequency F, Total iterations per epoch T, Binary mask $M \in \{0,1\}^N$, Dense network $W \in \mathbb{R}^N$, Pruned network $\overline{W} = M \odot W$
Initialize: Model weights W
[Sparse Structure Exploration phase]

```
Warming-up for 1 epoch
for epoch = 2, ..., E_e do
    calculate gradual sparsity p
    for Iter = 1, ..., T do
        if Iter % F == 0 then
            calculate pruning mask M with Eq. 1, 2, 3
            update sparse weights W̄ = W ⊙ M
        update weights W with gradient descent Eq. 5
    end for
end for
[Model compression by deleting zero weights]
[ Re-optimization Phase ]
for epoch = 1, ..., E_o do
    Pruning Network training
    update weights W̄ with gradient descent Eq. 10
end for
```

4.1 MAIN RESULT

Table 1 demonstrates the superior performance of our proposed SERo method on the ImageNet-1K dataset. While existing pruning methods struggled to maintain performance with higher pruning ratios, our Sparse Structure Exploration and Re-optimization (SERo) approach shows remarkable efficiency. For DeiT-Base model, SERo achieves approximately 69% reduction in computational cost (GFLOPs) with only 1.55% accuracy drop compared to the dense model. Notably, when compared to SNP [Shim et al., 2025], which achieved the highest pruning ratio of about 63% among existing methods, SERo achieves 6% more pruning while maintaining 0.62% higher accuracy.

For the smaller DeiT-Small model, SERo with 39% pruning achieving higher accuracy while pruning 5% more compared to SSVITE [Chen et al., 2021], which previously showed the best performance. SERo shows only 0.55% accuracy drop compared to the dense model. Furthermore, when compared to SNP, which had the highest compression ratio, SERo achieves similar pruning levels while showing 1.52% higher accuracy.

For DeiT-Tiny, SERo's effective structure exploration ca-

Table 1: Evaluation of pruning methods for DeiT models on ImageNet-1K dataset: accuracy (%), computational cost (GFLOPs), and number of parameters (M).

	Method	Top-1 (%)	Top-5 (%)	GFLOPs	Params (M)
DeiT-Tiny	Original [Touvron et al., 2021]	72.20	91.10	1.3	5.7
	SSViTE [Chen et al., 2021]	70.12	-	0.9	4.2
	WDPruning [Yu et al., 2022a]	70.34	89.82	0.7	3.5
	X-Pruner [Yu and Xiang, 2023]	71.10	90.11	0.6	-
	UVC [Yu et al., 2022b]	70.60	-	0.5	-
	SNP [Shim et al., 2025]	70.29	90.01	0.6	3.0
	SERo (ours)	**72.30**	**91.0**	**0.8**	**3.4**
DeiT-Small	Original [Touvron et al., 2021]	79.85	95.00	4.6	22.1
	SSViTE [Chen et al., 2021]	79.22	-	3.1	14.6
	WDPruning [Yu et al., 2022a]	78.38	94.05	2.6	13.3
	X-Pruner [Yu and Xiang, 2023]	78.93	94.24	2.4	-
	UVC [Yu et al., 2022b]	78.82	-	2.3	-
	SNP [Shim et al., 2025]	78.52	94.37	2.0	10.0
	SNP [Shim et al., 2025]	73.32	91.66	1.3	6.4
	SERo (ours)	**79.30**	**94.60**	**2.8**	**13.5**
	SERo (ours)	**74.84**	**92.42**	**1.5**	**7.2**
DeiT-Base	Original [Touvron et al., 2021]	81.80	95.59	17.6	86.6
	SSViTE [Chen et al., 2021]	82.22	-	11.8	56.8
	WDPruning [Yu et al., 2022a]	80.76	95.36	9.9	55.3
	X-Pruner [Yu and Xiang, 2023]	81.02	95.38	8.5	-
	UVC [Yu et al., 2022b]	80.57	-	8.0	-
	SNP [Shim et al., 2025]	79.63	94.37	6.4	31.6
	SERo (ours)	**80.25**	**94.98**	**5.4**	**27.0**

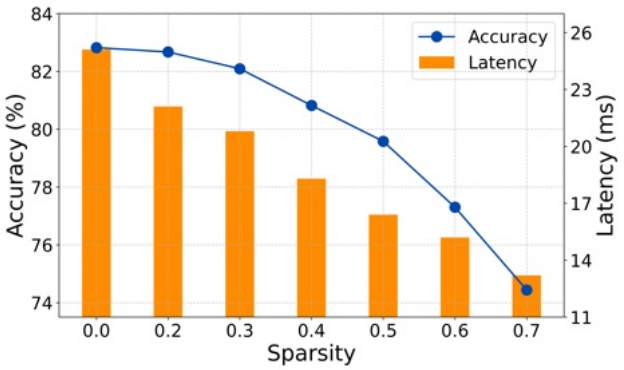


Figure 3: Relationship between model accuracy and inference latency across different sparsity levels.

pability has also been proven. Despite applying about 39% pruning, it achieved a 0.08% performance improvement compared to the unpruned original model. This demonstrates that SERo can improve performance through exploring and re-optimizing more efficient network structures beyond simple pruning. These results validate SERo's effective structure exploration and optimization capabilities across different model scales. Additional experimental results and detailed analysis can be found in Appendix.

4.2 COMPUTATIONAL EFFICIENCY

Table 2 compares the performance of various compression approaches applied to DeiT-base model trained with SERo.

Table 2: Performance comparison of various compression methods on DeiT-base model using ImageNet dataset, averaged over 1,000 iterations with batch size 64 on NVIDIA A6000 GPU (excluding the first 200 iterations as warm-up epochs). Here, dense represents the original model, Zeroing indicates the model with pruned weights replaced by zeros, while FFN Compression, Attention Compression, and All Compression refer to compression applied to Feed-Forward Network blocks only, Self-Attention blocks only, and the entire network architecture, respectively.

	Performance Metrics				
Metric	**Dense**	**Zeroing**	**FFN Compression**	**Attention Compression**	**All Compression**
Throughput (fps)	401.4	410.8 (1.0×)	652.6 (1.6×)	514.3 (1.3×)	**960.3** (2.4×)
Latency (ms)	159.5	155.8 (1.0×)	98.1 (1.6×)	124.5 (1.3×)	**66.6** (2.4×)
GFLOPs	17.6	17.6 (+0.0%)	9.3 (-47.2%)	13.7 (-22.2%)	**5.4** (-69.3%)
Parameters (M)	86.6	86.6 (+0.0%)	44.5 (-48.6%)	69.1 (-20.2%)	**27.0** (-68.8%)

The experimental results show that the Zeroing model maintains the same number of parameters and GFLOPs as the dense model, with similar processing speed. FFN Compression achieved a 48.6% reduction in parameters while increasing processing speed by 1.6 times, while Attention Compression resulted in a 20.2% parameter reduction with a 1.3 times speed improvement. Finally, the fully compressed model (All Compression) demonstrated the most significant improvements, reducing parameters by 68.8% while achieving a 2.4 times speed increase.

To further analyze the computational efficiency, we evalu-

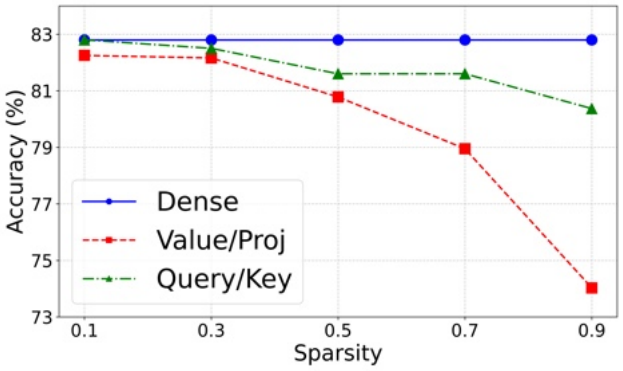


Figure 4: Accuracy comparison between the dense model and selective attention component pruning (Value & Proj vs. Query & Key) at varying sparsity levels.

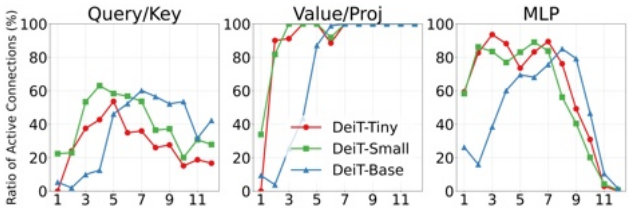


Figure 5: Unit-wise analysis per block of active connection ratios (%) (1 - pruning rate (%)) in DeiT models using SERo pruning: Comparison of Query & Key, Value & Proj, and FFN unit across DeiT-Tiny, Small, and Base variants at 0.4 sparsity level.

ated the relationship between accuracy and latency across different sparsity levels using DeiT-Tiny model trained on CIFAR-100 dataset, as shown in Figure 3. The results reveal that while latency decreases linearly with increased sparsity, model accuracy remains stable up to 0.3 sparsity before showing significant degradation beyond 0.4.

4.3 UNIT-WISE ANALYSIS

In this section, we analyze in detail how each pruning unit (Q&K, V&proj, FFN) as stated in Section 3.2 affects the performance of DeiT model pruned with SERo.

4.3.1 Comparative analysis of unit-wise pruning

Figure 4 compares the performance between the dense model and two pruning cases: pruning only the Value & Proj unit and pruning only the Query & Key unit. The experimental results show that pruning only the Value & Proj unit leads to performance degradation when sparsity exceeds 0.3, while pruning only the Query & Key unit maintains stable performance even at relatively high sparsity levels. This suggests that the Value & Proj unit plays a more crucial role in maintaining model performance.

Table 3: Comparison of sparse structures, mask similarities, and accuracies among different pruning methods (weight magnitude, gradient magnitude, and SERo) on DeiT-Tiny model with CIFAR-100 dataset.

	Pruning ratio (%) & Accuracy (%)			
	Unit Type	**Weight mag**	**Gradient mag**	**SERo (Ours)**
Sparsity=0.3	Query & Key	20.31	34.51	31.9
	Value & Proj	11.81	8.20	**4.82**
	FFN	36.97	34.32	35.82
	Accuracy	79.97	81.38	**82.09**
Sparsity=0.5	Query & Key	28.04	70.06	64.63
	Value & Proj	28.86	17.06	**12.54**
	FFN	60.77	53.07	55.71
	Accuracy	75.35	78.99	**79.58**
Sparsity=0.7	Query & Key	45.27	91.84	89.24
	Value & Proj	57.64	35.76	**25.30**
	FFN	79.28	73.10	76.37
	Accuracy	65.29	73.49	**74.25**
	Mask Similarity (%)			
Sparsity	**Sparsity=0.3**	**Sparsity=0.5**	**Sparsity=0.7**	
Gradient vs SERo	0.94	0.91	0.92	

4.3.2 Sparse structural analysis of active connections

To analyze the structural cause of this phenomenon, Figure 5 shows the active connection ratios (1 - pruning rate (%)) for each block at 0.4 sparsity level of DeiT model. Comparing the active connection patterns across Query & Key, Value & Proj, and FFN units, we observe that the Value & Proj unit consistently maintains a high ratio of active connections. Notably, in the middle blocks (4-8), the Value & Proj unit maintains over 90% of active connections.

Table 3 shows sparse structures and accuracy at various sparsity levels (0.3, 0.5, 0.7). A notable observation is the difference in the Value & Proj unit. SERo achieves higher accuracy while preserving more weights in the Value & Proj unit across all sparsity levels (12.54% at sparsity 0.5). Additionally, we conducted mask similarity analysis by training both gradient-based methods and SERo under identical initialization conditions and environments. Despite showing high similarity (over 90%), there are significant performance differences between them, suggesting that SERo finds more optimal network structures compared to conventional gradient magnitude methods.

4.3.3 Model-specific connection pattern analysis

For more detailed analysis, Figure 6 shows the block-wise active connection ratios in DeiT-Small model trained on the ImageNet dataset. While the FFN unit maintains relatively low active connection ratios in both early and later blocks, the Value & Proj unit maintains high active connection ratios across all blocks. This suggests that the Value & Proj unit plays a crucial role in maintaining model perfor-

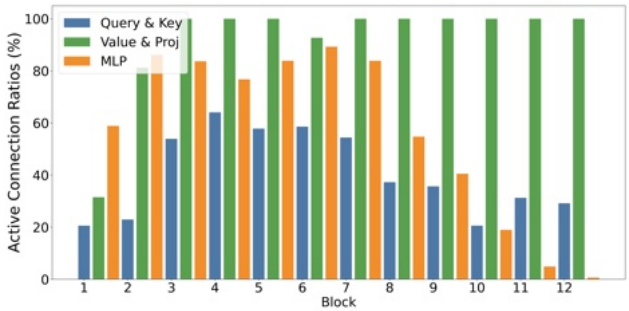


Figure 6: Active connection ratios per block of DeiT-Small (40% Sparsity) trained on ImageNet dataset.

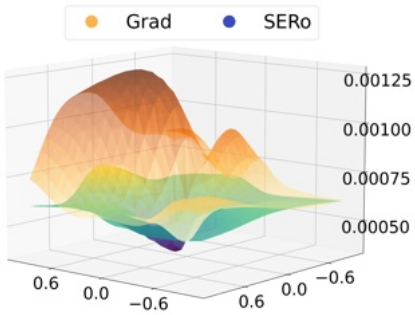


Figure 7: Loss landscapes by PyHessian [Yao et al., 2020] between gradient Mgnitude pruning (Orange) and SERo (Blue, Ours).

mance. These analysis results experimentally demonstrate the importance of the Value & Proj unit in transformer architecture and indicate that unit-wise characteristics should be considered when establishing effective pruning strategies.

4.4 METHOD VALIDATION

4.4.1 Loss landscape analysis

The correlation between loss landscape flatness and model generalization performance has been demonstrated in several studies [Hochreiter and Schmidhuber, 1997, Keskar et al., 2016]. In this section, we conducted loss landscape analysis to compare the performance of our proposed method with conventional gradient magnitude pruning. Specifically, after re-optimizing the structures found by each method, we visualized the loss landscapes using PyHessian [Yao et al., 2020]. The experiments were conducted on the CIFAR-100 dataset using DeiT-Tiny model. As shown in Figure 7, the model obtained using our proposed method exhibits a flatter loss landscape, suggesting better generalization performance.

Table 4: Accuracy comparison of different weight initialization methods (DeiT-Tiny model on CIFAR-100 with 50% Sparsity). No Init refers to using compressed parameters directly without re-initialization, preserving the original weight distribution.

	Initialization Methods		
Metric	**No Init**	**Gaussian**	**Xavier**
Top-1 Accuracy (%)	**79.58**	48.34	45.55

Table 5: Comparison of model accuracies between exploration and exploration + re-optimization phases of SERo on ImageNet dataset across different ViT architectures.

			Accuracy (%) on ImageNet	
	Model	**Sparsity**	**Exploration**	**Exploration + Re-optimization**
SERo	DeiT-Tiny	40%	68.14	**72.30**
	DeiT-Small	70%	69.18	**74.84**
	DeiT-Base	70%	77.33	**80.25**

4.4.2 Weight initialization impact and Re-optimization

To evaluate the effect of re-optimization on the weights corresponding to the structure found in Theorem 1, we compared various initialization methods. Table 4 analyzes the impact of different weight initialization methods after structure exploration and compression. As demonstrated in Eq. 11, The experimental results show that the weight inheritance (No Init; $W_1^{\overline{W}}$) achieves the highest accuracy at 79.58%, significantly outperforming both Gaussian [He et al., 2015] and Xavier [Glorot and Bengio, 2010] initialization methods. This suggests the importance of maintaining weights during the re-optimization process. Table 5 presents the results for No Init (Exploration), where re-optimization was not performed after exploration. It demonstrates that after finding an optimal architecture, optimizing the model after exploration in the optimized architecture (exploration + Re-optimization) is more efficient.

5 CONCLUSION

In this paper, we propose a Sparse Structure Exploration and Optimization (SERo), a novel pruning framework for vision transformers. Our approach presents a simple yet effective method that systematically analyzes the properties of pre-trained models to explore efficient sparse structures. In particular, instead of the conventional pruning approach of setting parameters to zero, we demonstrate significant improvements in computational costs and the inference speed by completely removing and compressing unnecessary pa-

rameters. Our experimental results show superior performance on various vision transformer models, suggesting that the proposed framework could be extended beyond vision tasks to other domains, including NLP tasks with transformer-based models.

Acknowledgements

This work was supported by Hyundai Motor Company and Kia, and partially supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (o.RS-2025-02219317, AI Star Fellowship(Kookmin University)).

References

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL https://arxiv.org/abs/1502.01852.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks, 2018.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4340–4349, 2019.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Flat Minima. *Neural Computation*, 9(1):1–42, 01 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.1.1. URL https://doi.org/10.1162/neco.1997.9.1.1.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Jangho Kim, Jayeon Yoo, Yeji Song, KiYoon Yoo, and Nojun Kwak. Finding efficient pruned network via refined gradients for pruned weights. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9003–9011, 2023.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation, 2020. URL https://arxiv.org/abs/1907.11922.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map, 2020a.

Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search, 2020b.

Tao Lin, Sebastian U Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. *International Conference on Learning Representations (ICLR)*, 2020c.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

Yinglu Liu, Hailin Shi, Hao Shen, Yue Si, Xiaobo Wang, and Tao Mei. A new dataset and boundary-attention semantic segmentation for face parsing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11637–11644, Apr. 2020. doi: 10.1609/aaai.v34i07.6832. URL https://ojs.aaai.org/index.php/AAAI/article/view/6832.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. URL https://arxiv.org/abs/2103.14030.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning, 2019. URL https://arxiv.org/abs/1810.05270.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild, 2015. URL https://arxiv.org/abs/1411.7766.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression, 2017.

Mansheej Paul, Feng Chen, Brett W Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? *International Conference on Learning Representations (ICLR)*, 2023.

Lixiong Qin, Mei Wang, Xuannan Liu, Yuhang Zhang, Wei Deng, Xiaoshuai Song, Weiran Xu, and Weihong Deng. Faceptor: A generalist model for face perception, 2024. URL https://arxiv.org/abs/2403.09500.

Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 397–403, 2013. doi: 10.1109/ICCVW.2013.59.

Soumyadip Sengupta, Jun-Cheng Chen, Carlos Castillo, Vishal M. Patel, Rama Chellappa, and David W. Jacobs. Frontal to profile face verification in the wild. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, 2016. doi: 10.1109/WACV.2016.7477558.

Kyunghwan Shim, Jaewoong Yun, and Shinkook Choi. Snp: Structured neuron-level pruning to preserve attention scores, 2024. URL https://arxiv.org/abs/2404.11630.

Kyunghwan Shim, Jaewoong Yun, and Shinkook Choi. Snp: Structured neuron-level pruning to preserve attention scores. In *European Conference on Computer Vision*, pages 90–104. Springer, 2025.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers distillation through attention, 2021. URL https://arxiv.org/abs/2012.12877.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Xianzhe Xu, Yiqi Jiang, Weihua Chen, Yilun Huang, Yuan Zhang, and Xiuyu Sun. Damo-yolo : A report on real-time object detection design, 2023. URL https://arxiv.org/abs/2211.15444.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE, 2020.

Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 2022, 2022a.

Lu Yu and Wei Xiang. X-pruner: explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24355–24363, 2023.

Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. *arXiv preprint arXiv:2203.08243*, 2022b.

Sixing Yu, Arya Mazaheri, and Ali Jannesari. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In *International conference on machine learning*, pages 25656–25667. PMLR, 2022c.

Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: pruning weights that cancel one another in neural networks. In *International Conference on Machine Learning*, pages 25668–25683. PMLR, 2022d.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder, 2017. URL https://arxiv.org/abs/1702.08423.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*, 2021.

Max Zimmer, Christoph Spiegel, and Sebastian Pokutta. How i learned to stop worrying and love retraining. *International Conference on Learning Representations (ICLR)*, 2023.

A CONVERGENCE ANALYSIS

The update rule of SERo in the exploration phase is as follows:

$$W_{t+1} = W_t - \eta \cdot \nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W^{|\cdot|}$$

Consider the squared distance between W_{t+1} and W_*:

$$\|W_{t+1} - W_*\|^2 = \|W_t - \eta \cdot \nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|} - W_*\|^2.$$

Taking the expectation conditioned on W_t:

$$\mathbb{E}[\|W_{t+1} - W_*\|^2 \mid W_t] = \|W_t - W_*\|^2 - 2\eta \mathbb{E}[\langle \nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|}, W_t - W_* \rangle \mid W_t] + \eta^2 \mathbb{E}[\|\nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|}\|^2 \mid W_t].$$

Using the assumption that $\mathbb{E}[\|\nabla_{\overline{W_t}} \mathcal{L}\|^2 \mid W_t] \leq G^2$ and noting that $\|a \cdot b\|^2 = \|a\|^2 \|b\|^2$ for element-wise multiplication, we have:

$$\mathbb{E}[\|\nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|}\|^2 \mid W_t] \leq G^2 \|M_t \cdot W_t^{|\cdot|}\|^2.$$

by convexity and the loss difference condition,

$$\langle \nabla_{\overline{W_t}} \mathcal{L} \cdot M_t \cdot W_t^{|\cdot|}, W_t - W_* \rangle \geq \mathcal{L}(\overline{W_t}) - \mathcal{L}(W_*) - \epsilon.$$

Plugging these bounds back:

$$\mathbb{E}[\|W_{t+1} - W_*\|^2 \mid W_t] \leq \|W_t - W_*\|^2 - 2\eta(\mathcal{L}(\overline{W_t}) - \mathcal{L}(W_*) - \epsilon) + \eta^2 G^2 \|M_t \cdot W_t^{|\cdot|}\|^2.$$

Summing over $t = 1$ to T, the left-hand side telescopes:

$$\|W_{T+1} - W_*\|^2 \leq \|W_1 - W_*\|^2 - 2\eta \sum_{t=1}^{T} (\mathcal{L}(\overline{W_t}) - \mathcal{L}(W_*) - \epsilon) + \eta^2 G^2 \sum_{t=1}^{T} \|M_t \cdot W_t^{|\cdot|}\|^2.$$

Rearranging:

$$\frac{1}{T} \sum_{t=1}^{T} \mathcal{L}(\overline{W_t}) - \mathcal{L}(W_*) \leq \frac{\|W_1 - W_*\|^2}{2\eta T} + 2\eta T \epsilon + \frac{\eta G^2}{2T} \sum_{t=1}^{T} \|M_t \cdot W_t^{|\cdot|}\|^2.$$

Taking the expectation over all randomness:

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} \mathcal{L}(\overline{W_t})\right] - \mathcal{L}(W_*) \leq \frac{\|W_1 - W_*\|^2}{2\eta T} + 2\eta T \epsilon + \frac{\eta G^2}{2} \mathbb{E}[\|M_t \cdot W_t^{|\cdot|}\|^2].$$

A.1 CONVERGENCE BOUND OF RE-OPTIMIZATION PHASE

With a fixed explored sparse structure, the convergence bound for standard gradient descent over T iterations is given by:

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} \mathcal{L}(W_t)\right] - \mathcal{L}(W_*) \leq \frac{\|W_1 - W_*\|^2}{2\eta T} + \frac{\eta G^2}{2}.$$

When initializing $\tilde{W}_1$ with the parameters found in the exploration phase ($W_1^{\overline{W}}$), the distance to the optimal solution is smaller compared to random initialization (W_1^{random}). Specifically, since:

$$\|W_1^{\text{random}} - W^*\|^2 > \|W_1^{\overline{W}} - W^*\|^2,$$

the improvement in the convergence bound is given by:

$$\Delta = \frac{\|W_1^{\text{random}} - W^*\|^2 - \|W_1^{\overline{W}} - W^*\|^2}{2\eta T}.$$

Thus, initializing from $W_1^{\overline{W}}$ leads to faster convergence compared to random initialization.

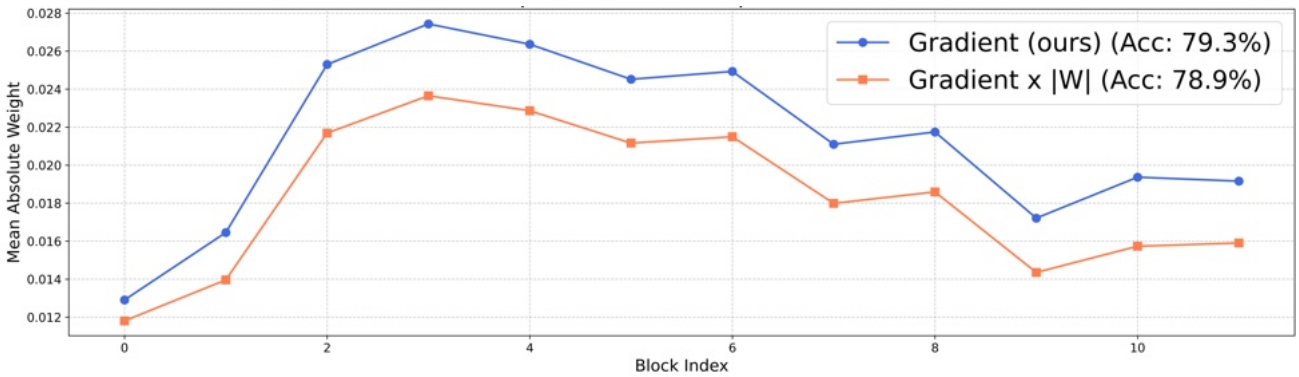


Figure 8: Mean absolute weights per block with different update methods in re-optimization phase using DeiT-Small model with 40% sparsity on ImageNet dataset

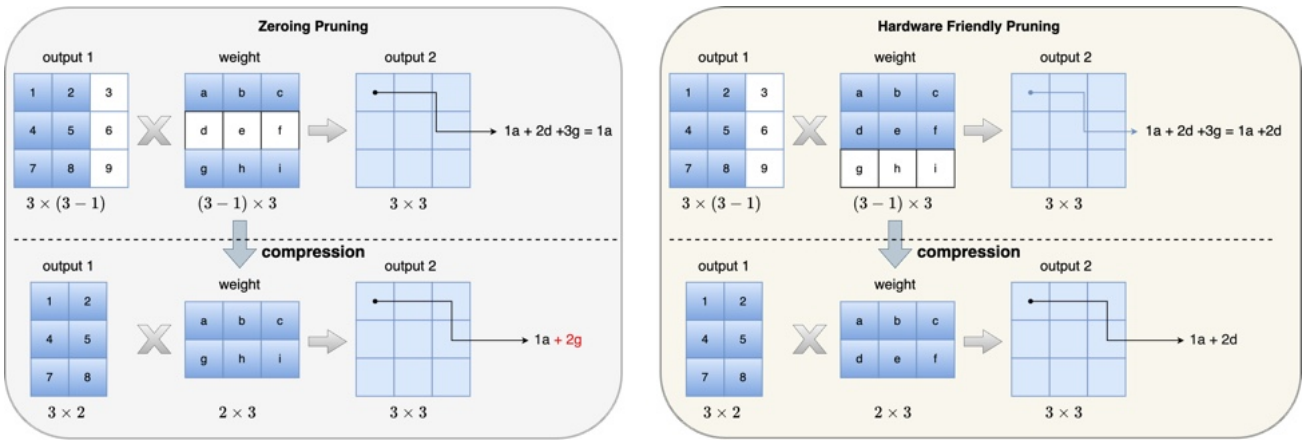


Figure 9: Illustration of dimension mismatch caused by non-shared weight masking (left) versus Hardware Friendly Pruning with consistent dimensions through shared masking (right).

B COMPARATIVE ANALYSIS OF GRADIENT UPDATE METHODS: STANDARD VS WEIGHT-SCALED APPROACHES

Figure 8 shows the average weights per block and accuracy after applying different gradient update methods in re-optimization phase. The blue line represents our standard gradient update method, while the orange line shows the $gradient \times |W|$ approach used in exploration phase. The results indicate that multiplying by $|W|$ leads to overall smaller weight values and lower accuracy. This occurs because $|W|$ is less than 1, effectively reducing the update magnitude. Such reduced updates essentially mimic a lower learning rate, potentially slowing down optimization or increasing the risk of getting trapped in local minima.

C HARDWARE FRIENDLY PRUNING

When masks are not shared across query/key, value/proj, and FFN layers, dimension mismatch problems similar to Zeroing Pruning may arise. The left side of Figure 9 illustrates that removing the third column from the input and the second row from the weight matrix results in unnecessary computations such as 1a + 2d + 3g in the final output. Upon actual compression of such a model, as shown in the bottom left, the computational results deviate from pre-compression values. This presents a significant obstacle to effective hardware optimization. Therefore, as demonstrated in the right image, we need a hardware-friendly pruning approach that maintains consistent input/output dimensions.

D ANALYSIS OF WEIGHT DISTRIBUTIONS

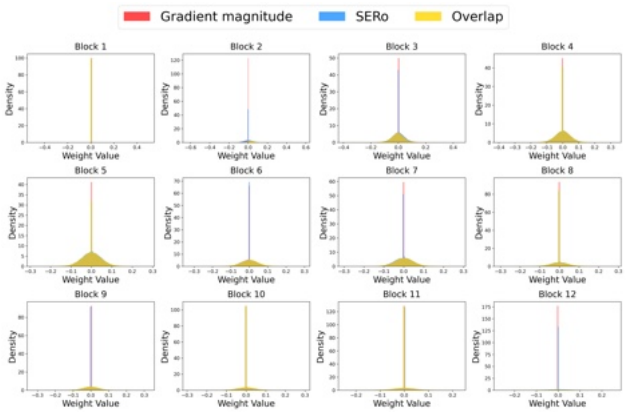


Figure 11: Weight distribution comparison of DeiT-Tiny model trained on CIFAR-100: Weight distributions per block of SERo (blue) and Gradient magnitude (red). yellow regions indicate overlapping distributions.

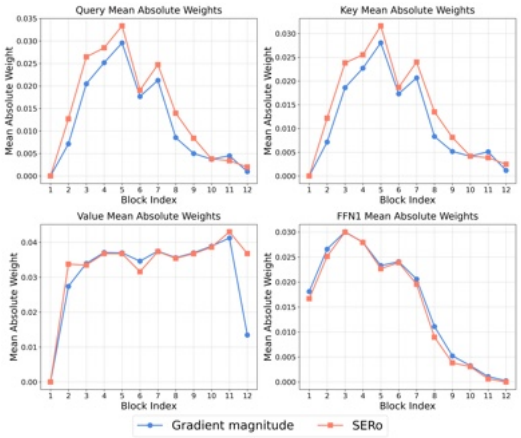


Figure 12: Weight average comparison of Query/Key/Value/FFN1 per block in DeiT-Tiny model trained on CIFAR-100: SERo (orange) and Gradient magnitude (blue)

As shown in Figure (10, 11), while SERo (blue) and Gradient magnitude (red) exhibit similar pruning patterns (yellow overlapping regions), SERo's superior performance can be attributed to subtle differences in non-overlapping regions and particularly effective weight preservation in the Value layer.

Figure D further supports this analysis, showing that Query and Key layers display very similar patterns with peaks at block 5 followed by gradual decrease, while the Value layer maintains consistently high mean weights after block 1, with a notable difference at block 12. SERo (orange) maintains higher mean absolute weight values throughout the network, indicating better preservation of important weights. Notably, the Value layer plays a particularly crucial role in this preservation, demonstrating SERo's effectiveness in maintaining essential weight information for model performance.

E IMPLEMENTATION DETAILS

We conduct experiments using the officially released DeiT models, maintaining consistent hyperparameters for both sparse structure exploration and re-optimization phases. The sparse structure exploration runs for 50 epochs and re-optimization for 150 epochs, with both phases using a learning rate of 5e-5 and a weight decay of 0.05. The AdamW optimizer with a cosine learning rate scheduler is applied in all experiments. The batch size is set to 256 or 512 for the ImageNet-1K dataset and 128 for CIFAR datasets. All ImageNet-1K experiments are conducted on NVIDIA A6000 GPU, while CIFAR experiments are run on NVIDIA RTX 3090 GPU. For more implementation details, please refer to our released code.

F MODEL AND TASK GENERALIZATION

To verify whether SERO's applicability is limited to classification tasks on DeiT models, we conducted experiments as shown in Tables 6, 7, and 8.

First, in Table 6, we performed classification task experiments on the Swin Transformer Liu et al. [2021] model rather than the DeiT model. As specified in the training details of the paper, we applied the hyperparameters used for DeiT training to perform 50% pruning on this model and measured the computational cost (FLOPs) and accuracy. The experimental results showed that despite significantly improving efficiency by reducing FLOPs by approximately 50%, the accuracy drop was only about 1.9%.

In Table 7, to evaluate the effectiveness of SERO in non-classification tasks such as object detection, we applied SERO to the DAMO Xu et al. [2023] model architecture, which incorporates attention mechanisms, and conducted evaluation on the COCO 2017 Lin et al. [2015] dataset. When SERO was applied to the DAMO-NL variant achieving approximately 48% parameter reduction, FLOPs decreased by 48.4% while mAP dropped by only 2.07 points (from 40.5 to 38.43). Similarly, other DAMO model variants (NS, NM) also showed only limited mAP performance degradation despite significant reductions in FLOPs and parameters.

Table 6: Comparison of Swin Transformer Models with SERo

Model	Parameters	Pruning Rate (%)	FLOPs	Accuracy
Swin-T (Original)	29M	-	4.5G	81.3%
Swin-S (Original)	50M	-	8.7G	83.0%
Swin-S + SERo	25M	50%	4.3G	81.1%

Table 7: Performance Comparison of DAMO Models with SERo

Model	Pruning Method	Compression Ratio	FLOPs (G)	Params (M)	mAP @ 0.5:0.95	Latency Orin batch 1	Latency Nuc batch 1
DAMO NS	Original	-	1.62	1.41	32.3	3.06	110.78
DAMO NS	SERo (ours)	33%	1.09	0.93	30.94	2.29	66.67
DAMO NM	Original	-	3.8	2.71	38.2	2.56	72.72
DAMO NM	SERo (ours)	49%	1.99	1.37	35.19	2.19	49.59
DAMO NL	Original	-	6.16	5.69	40.5	1.90	38.05
DAMO NL	SERo (ours)	48%	3.18	2.87	38.43	1.61	28.9

Table 8: Performance Comparison of Faceptor Models with SERo

Model	Parameters (M)	FLOPs (G)	CFP_FP (Val Acc ↑)	UTK_FACE (MAE ↓)	CelebA (mACC)	CelebAMask-HQ (F1-mean ↑)	LaPa (F1-mean ↑)	LaPa (Inter-Ocular ↓)	300W (Inter-Pupil ↓)
Faceptor	105.2	108.46	96.14	4.21	23.12	88.22	91.94	4.63	6.67
SERo	62.59	67.55	96.46	4.32	23.13	88.00	91.69	4.52	6.51
SERo	42.57	46.97	92.69	4.41	23.13	86.74	90.95	4.68	6.74

Additionally, Table 8 shows the experimental results on the Faceptor Qin et al. [2024] model, a more complex transformer model that combines encoder-decoder architecture, using six datasets Sengupta et al. [2016], Zhang et al. [2017], Liu et al. [2015], Lee et al. [2020], Liu et al. [2020], Sagonas et al. [2013]. SERO maintained strong performance across various face-related tasks even at high compression rates (achieving 62.59M parameters with approximately 40% compression), with minimal performance degradation and even performance improvements in some tasks such as Face ID (CFP_FP) and Face Alignment (300W, LaPa).

G HYPERPARAMETER SENSITIVITY ANALYSIS

In this section, we conduct an analysis of hyperparameter sensitivity. These studies utilized the DeiT-small model pruned to 70% sparsity, with performance evaluated based on CIFAR-100 accuracy.

Table 9 examines the effect of learning rate in the re-optimization phase. When set higher than the exploration phase learning rate (5e-4 vs. 5e-5), convergence problems occurred, resulting in a low accuracy of 78.8%. In contrast, using the same learning rate (5e-5) for both exploration and re-optimization achieved the best performance of 83.1% accuracy. This is because it can effectively maintain the sparse structure discovered during training. Using a lower learning rate (5e-6) than the exploration phase resulted in insufficient optimization, achieving only 80.69% performance. Therefore, we confirmed that using the same learning rate for both phases is the most effective approach.

Table 10 examines the impact of pruning mask update frequency on model performance. The frequency of 8 (mask update every 8 epochs) used in the main experiments achieved the best performance of 83.1%. When updating the mask every epoch (frequency=1), performance decreased to 82.24% due to structural instability. In contrast, lower update frequencies (frequency 16, 32, 64) maintained stable performance in the range of 82.7 82.9%. This demonstrates that excessively frequent mask updates negatively affect performance, while stability is secured above a certain threshold.

Table 11 analyzes the impact of epoch allocation between the exploration and re-optimization phases on performance. When configured with 50 epochs for exploration and 150 epochs for re-optimization, the optimal balance was achieved, recording

the highest accuracy of 83.1%. This demonstrates the importance of maintaining an appropriate balance between exploration and re-optimization. When exploration was insufficient (2 epochs), performance was limited to 82.4%, while conversely, excessive exploration with minimal re-optimization (199 exploration epochs, 1 re-optimization epoch) significantly degraded performance to 78.8%.

Table 9: Effect of learning rate in the re-optimization phase on model performance. DeiT-small model pruned to 70% sparsity evaluated on CIFAR-100.

	Step	LR	Acc (%)
SERo	Exploration	5e-5	79.67
	Exploration + Re-optimization	5e-4	78.8
	Exploration + Re-optimization	5e-5	**83.1**
	Exploration + Re-optimization	5e-6	80.69

Table 10: Impact of pruning mask update frequency on model performance. DeiT-small model pruned to 70% sparsity evaluated on CIFAR-100.

	Frequency	Acc (%)
SERo	1	82.24
	8	**83.1**
	16	82.75
	32	82.71
	64	82.9

Table 11: Impact of epoch allocation between exploration and re-optimization phases on model performance. DeiT-small model pruned to 70% sparsity evaluated on CIFAR-100.

	Exploration	Re-optimization	Accuracy (%)
SERo	2	198	82.4
	25	175	82.9
	50	150	**83.1**
	100	100	82.5
	150	50	82.39
	175	25	81.85
	199	1	78.8

As shown in Table 12, SERo significantly reduces the computational operations in the re-optimization phase compared to the fine-tuning phase. Notably, for the DeiT-Base model with 70% sparsity, the computational operations (GFLOPs) decrease from 17.6 to 5.4, showing approximately 70% reduction, which demonstrates that our proposed re-optimization strategy can effectively alleviate the computational burden of the model. Furthermore, Table 13 demonstrates that global pruning consistently outperforms layer-wise pruning across all model variants on CIFAR-100. The performance gap is particularly notable for DeiT-Small, where global pruning achieves 83.34% accuracy compared to 78.28% with layer-wise pruning, suggesting that maintaining the flexibility to redistribute sparsity across layers is beneficial for model performance.

Table 12: Comparison of GFLOPs between fine-tuning and re-optimization phases of SERo across different DeiT architectures.

			GFLOPs	
	Model	**Sparsity**	**Fine-tuning**	**Re-optimization**
SERo	DeiT-Tiny	40%	1.3	**0.8**
	DeiT-Small	70%	4.6	**1.5**
	DeiT-Base	70%	17.6	**5.4**

Table 13: Comparison of model accuracies between layer-wise and global pruning strategies of SERo on CIFAR-100 dataset (Sparsity = 50%).

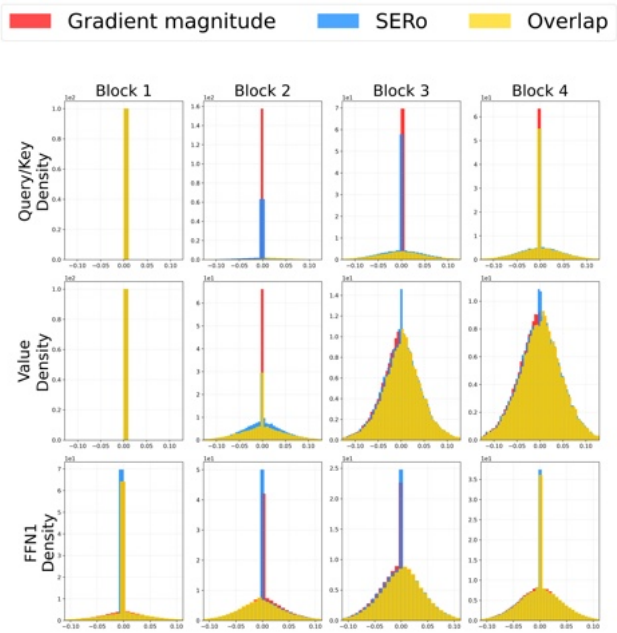
	Accuracy (%) on CIFAR-100			
	Model	**Sparsity**	**Layer-wise**	**Global**
SERo	DeiT-Tiny	50%	76.95	**79.58**
	DeiT-Small	50%	78.28	**83.34**
	DeiT-Base	50%	83.02	**85.76**

H ATTENTION PATTERN ANALYSIS BEFORE AND AFTER PRUNING

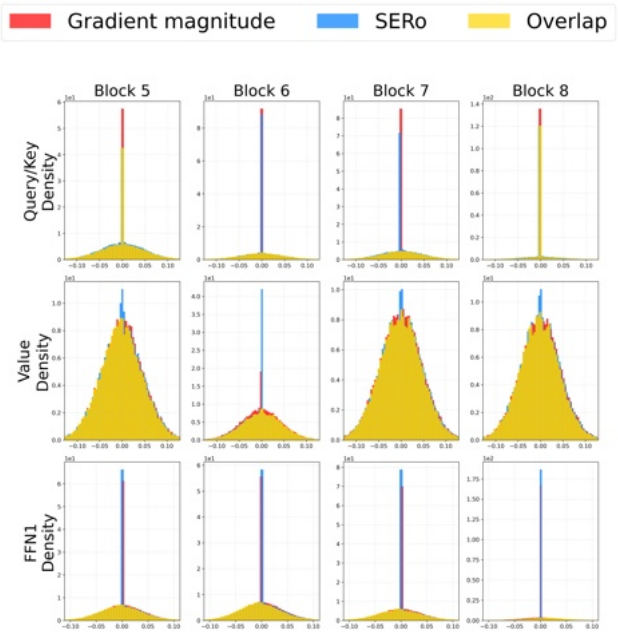
Figure 13 shows the per-block GradCAM visualization comparing the original and pruned models. The pruned model demonstrates reduced overall noise and shows improved focus on key features such as the chimpanzee's face and form. This

suggests that the pruning process effectively preserved the model's essential feature extraction capabilities while eliminating unnecessary activations.

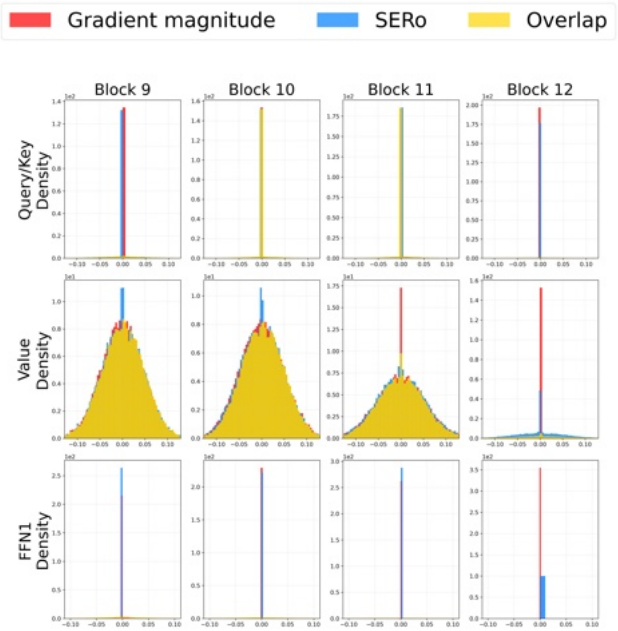
The attention map analysis at blocks 4 and 10 Figure (14, 15) reveals interesting pattern differences. In Figure 14, the original model shows distributed attention patterns across wider regions, while the pruned model exhibits more focused attention on key features of the peacock (head, tail). In Figure 15, both models demonstrate sparse attention patterns, reflecting the tendency to focus on specific feature points in higher blocks. These findings demonstrate that model compression can achieve efficient processing while maintaining key feature recognition capabilities.



(a) Block (1 4) distribution



(b) Block (5 8) distribution



(c) Block (9 12) distribution

Figure 10: Weight distribution comparison of DeiT-Tiny model trained on CIFAR-100: block-wise Query/Key/Value/FFN1 distributions of SERo (blue) and Gradient magnitude (red). yellow regions indicate overlapping distributions.

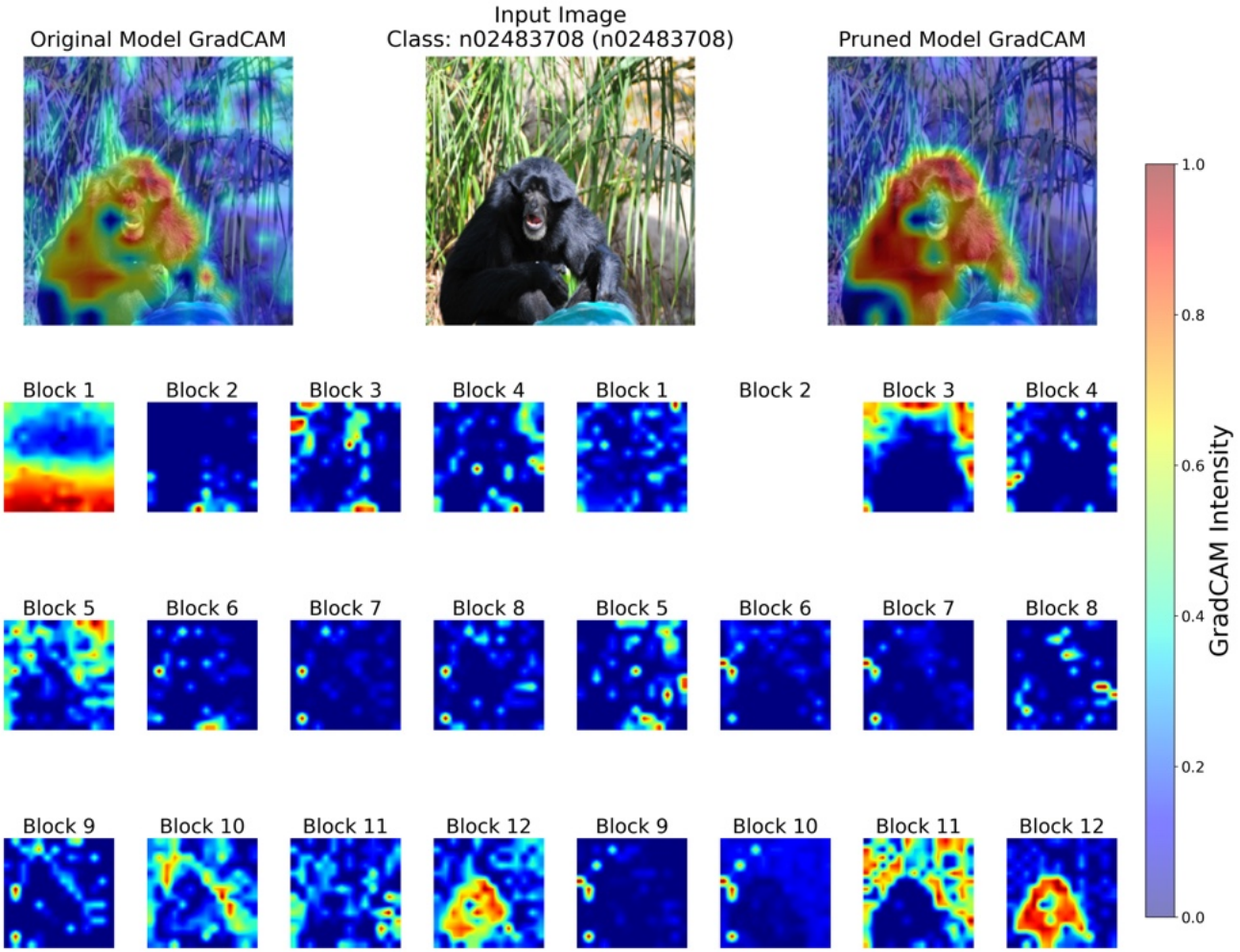


Figure 13: Comparison of combined and individual block GradCAM between original(left) and 70% pruned(right) DeiT-Base model

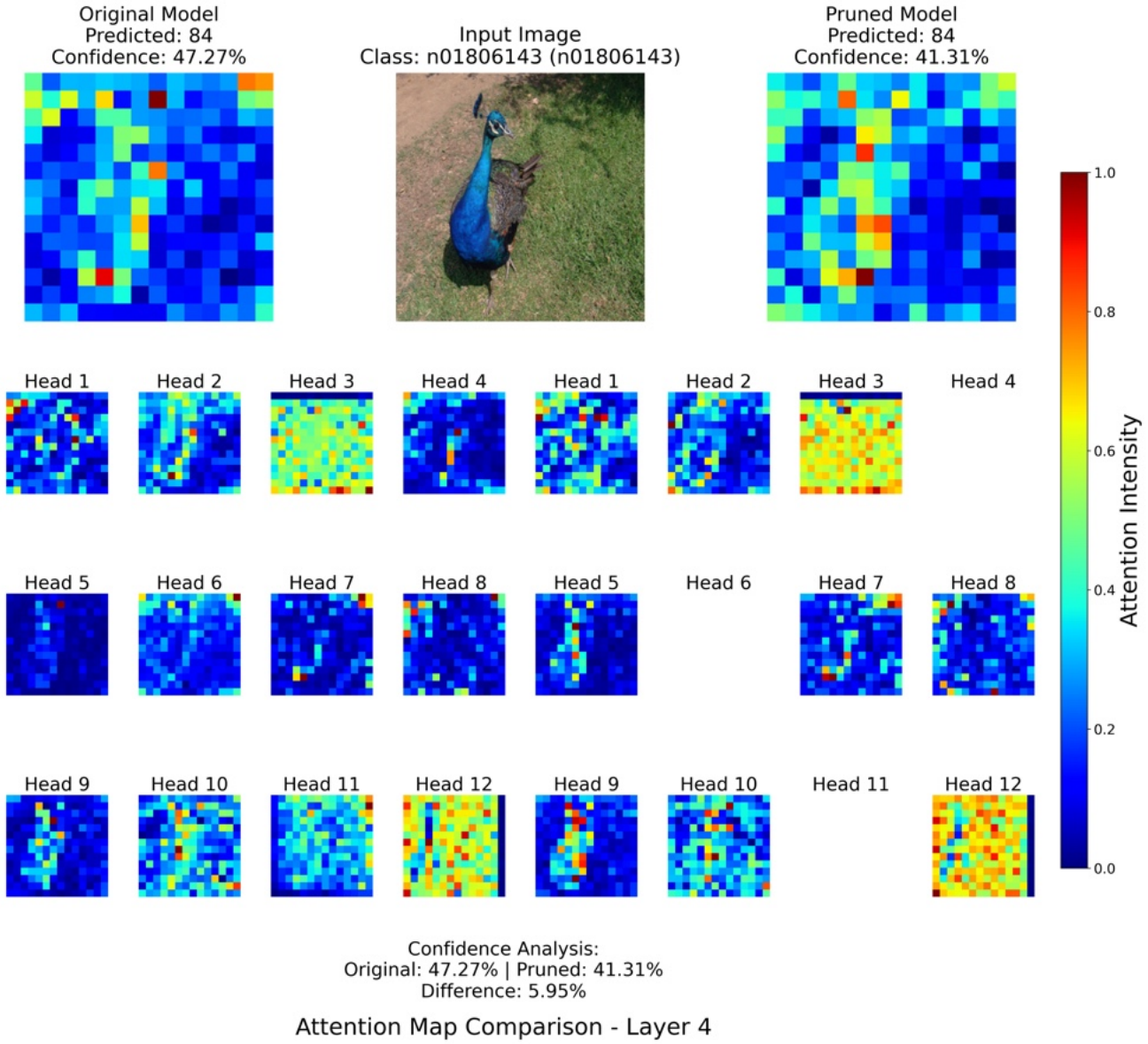


Figure 14: Comparison of combined and individual head attention maps between original(left) and 70% pruned(right) DeiT-Base model at block 4

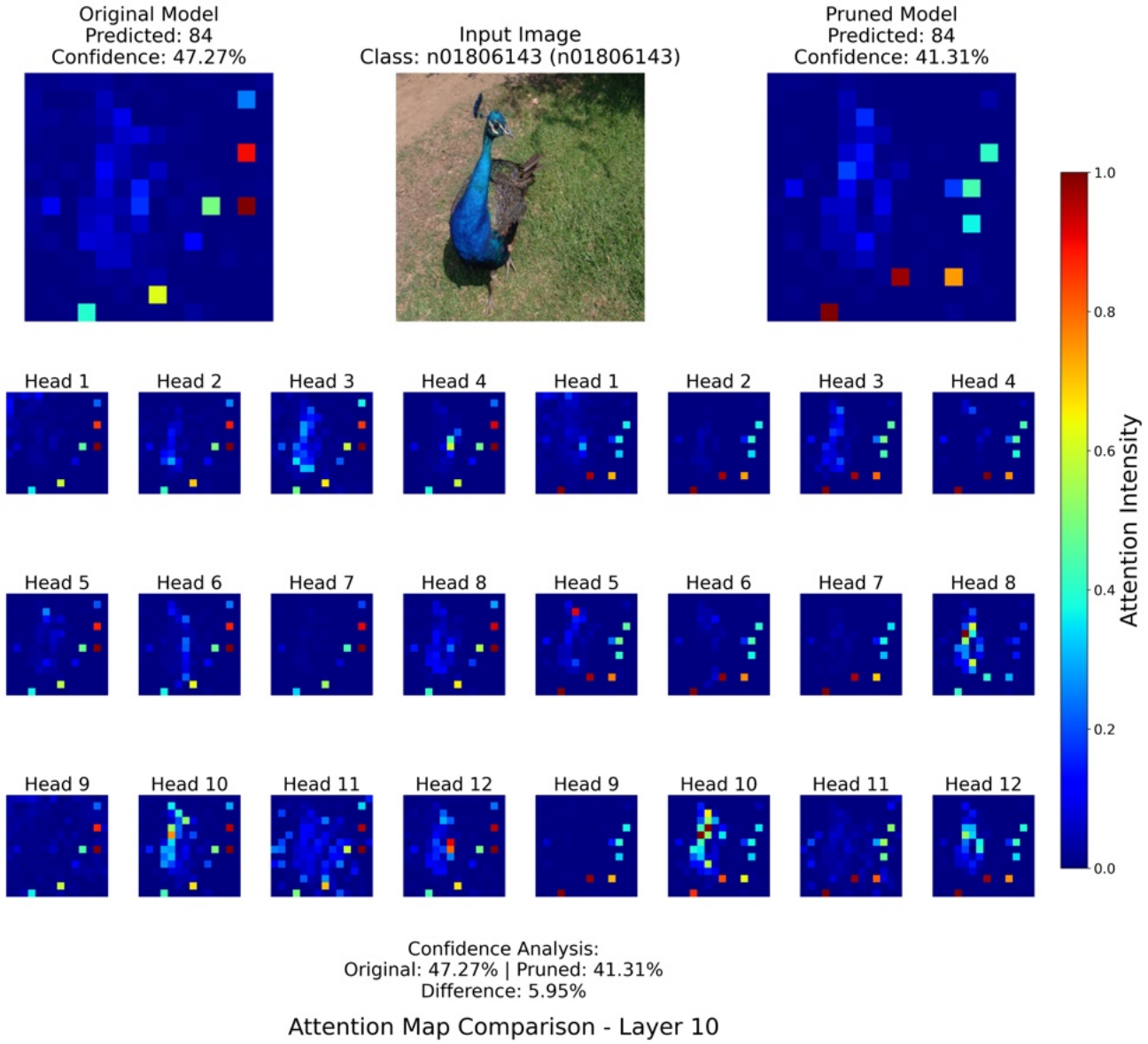


Figure 15: Comparison of combined and individual head attention maps between original(left) and 70% pruned(right) DeiT-Base model at block 10