SlimSwin: Gradient-Based Window-Level Head Pruning for Efficient Vision Transformers

Emir Mehmet Eryılmaz
Computer Science Department
Ozyegin University
Istanbul, Türkiye
emir.eryilmaz@ozu.edu.tr

Ismail Akturk
Computer Science Department
Ozyegin University
Istanbul, Türkiye
ismail.akturk@ozyegin.edu.tr

***Abstract*—Vision Transformers (ViTs) achieve competitive performance across vision tasks, but their global self-attention mechanism is computationally expensive due to quadratic complexity. Window-based or local-attention variants reduce this cost to near-linear complexity, although many attention heads in these models remain redundant and contribute little to the final prediction. In this work, we propose a window-level attention head pruning method that exploits the observation that only a subset of heads within each window is truly informative. Unlike existing magnitude-based pruning strategies, we introduce a gradient-based importance estimation to measure each head's contribution to the loss, enabling fine-grained removal of redundant or unhelpful heads. Our approach preserves accuracy while significantly reducing computation. Experiments on ImageNet-1K demonstrate that SlimSwin achieves significant MAC reductions up to around 60% in the attention blocks while incurring at most about a 1.5% drop in top-1 accuracy at the highest pruning levels, providing an effective avenue for improving the efficiency of window-based vision transformers.**

***Index Terms*—Vision Transformers, attention head pruning, window-based attention, Swin Transformer, model compression, efficient inference**

I. Introduction

Convolutional neural networks (CNNs) have been central to a wide range of computer vision tasks, including image classification [1]–[3], object detection [4], [5], and object segmentation [6], [7], following the breakthrough of AlexNet [8]. In parallel, the development of the self-attention mechanism [9] drove substantial progress in natural language processing. Although several studies have investigated integrating self-attention into vision architectures [10]–[13], self-attention-based models did not become competitive alternatives to CNNs until the introduction of the Vision Transformer (ViT) by Dosovitskiy et al. [14]. However, ViT exhibited two major limitations. First, it was highly data-inefficient, requiring large-scale datasets to achieve competitive performance. Second, its computational complexity increased quadratically with image resolution, making it unsuitable for processing high-resolution images. While the data-inefficiency issue was largely addressed through improved training strategies and the use of knowledge distillation with minimal architectural modifications [15], the resolution-scaling problem required substantial architectural changes. To address this limitation, Liu et al. [16] introduced the Swin Transformer, which employs local self-attention instead of global attention. The core idea is to partition feature maps into windows and compute attention only among patches within each window. To capture cross-window interactions, neighboring windows are merged hierarchically after each stage, leading to a single window at the final stage. The Swin Transformer not only scales linearly with image resolution, but also outperformed contemporary ViT models. Nevertheless, like other attention-based architectures, it still incurs considerable computational cost.

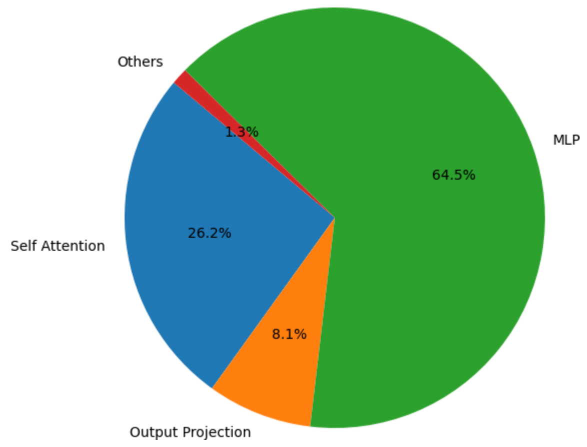

Fig. 1. Component-wise FLOPs Breakdown of the Swin Transformer

Figure 1 shows the distribution of FLOPs across the different components of the transformer. Although MLP layers appear to dominate the overall computational cost, the compression of MLPs and convolutional layers is a well-studied topic [17]–[24]. Furthermore, as shown in Figure 1, self-attention and output projection also account for a substantial proportion of the total FLOPs. Applying existing weight-compression techniques to self-attention is not straightforward, since the query, key, and value matrices are input-dependent, rather than fixed parameters as in MLP layers. Therefore, an alternative approach tailored to the characteristics of self-attention is required.

Several recent works have attempted to reduce the computational complexity of self-attention in vision models. Rao et al. [25] and Tang et al. [26] proposed pruning image patches based on their estimated importance, achieving significant sparsity with minimal accuracy degradation. However, because these methods rely on fine-grained patch pruning, the resulting sparsity patterns are difficult to exploit efficiently on existing hardware. In contrast, Chen et al. [27] introduced a coarse-grained strategy that prunes entire windows in the Swin Transformer, though this eliminates fine-grained control and risks discarding informative regions along with uninformative ones. These limitations highlight the need for pruning strategies that reduce attention complexity while preserving useful structural information.

In this paper, we propose a window-wise head pruning approach based on the hypothesis that, for a given window, only a subset of attention heads contributes meaningfully while others are redundant or uninformative. A key challenge is accurately estimating head importance so that less informative heads can be removed without degrading accuracy. Whereas prior work [26], [27] typically relies on magnitude-based importance measures, our method employs a gradient-based criterion to obtain a more reliable estimate of each head's contribution.

The rest of the paper is organized as follows. Section II reviews prior work related to our approach and highlights the key differences between these methods and ours. Section III provides background on the Swin Transformer, multi-head self-attention, head redundancy, and the computational cost of attention. Section IV introduces our methodology by explaining how head importance is determined along with our pruning algorithm. Section V presents the experimental setup, including model configurations, training scheme, and implementation details. Section VI reports and discusses the results, and Section VII concludes the paper.

II. Related Works

A. Attention Head Pruning

Michel et al. [28] conducted one of the primary studies showing that many attention heads can be removed with minimal impact on Transformer accuracy. They demonstrated that some layers can even be reduced to a single head, and that moderate head pruning may improve accuracy in tasks, such as machine translation. Their pruning criterion is gradient-based, obtained by applying a head-wise binary mask to the attention matrix and computing the gradient of this mask. While we also employ a gradient-based importance metric, our approach differs by estimating importance jointly for each window-head pair rather than assuming that heads are uniformly important or unimportant across all tokens.

Voita et al. [29] similarly showed that a large fraction of attention heads can be removed without degrading translation quality and provided an in-depth analysis of the functional roles of individual heads. Using layer-wise relevance propagation [30], they quantified how much each head contributes to the final output logits at different layers, reinforcing the idea that not all heads are equally valuable.

Wang et al. [31] proposed a unified framework combining token pruning, head pruning, and quantization to accelerate Transformer inference. Their method relies on attention probabilities as importance scores and adopts a pruning pipeline in which any token or head removed at one layer is permanently eliminated in subsequent layers. They further enhance efficiency through progressive quantization, in which attention probabilities are initially computed using only the most significant bits.

Although these works achieve notable efficiency gains, they share two key limitations. First, they treat head importance as global, despite the evidence that the relevance of a head may vary significantly across different tokens. This suggests that a token-dependent or context-dependent head pruning mechanism could more effectively preserve essential computations. Second, these methods have been developed exclusively for NLP tasks, and analogous strategies have not been thoroughly explored in the vision domain, particularly in architectures, such as Swin Transformer where locality and window structure play central roles. Motivated by these gaps, we propose a window-wise head pruning approach that estimates head importance separately within each window. This formulation captures the context-dependent utility of attention heads, while aligning naturally with the hierarchical design of vision transformers.

B. Pruning in Vision Transformers

Tang et al. [26] proposed a top-down patch pruning procedure for vision transformers. Their method begins by retaining only the classification token in the final layer and progressively reintroducing patches in earlier layers by preserving only the most important ones. To rank patches, they introduce a patch saliency metric based on Lipschitz continuity [32], [33], which estimates the contribution of each patch in a given layer to the final output.

Rao et al. [25] introduced a dynamic patch pruning approach in which the set of retained patches is determined at run time by a prediction module. The module uses MLPs to extract global and local features and outputs per-token probabilities indicating which tokens should be preserved or pruned. To enable differentiable sampling from these probabilities, they adopt the Gumbel-Softmax technique [34], allowing end-to-end training.

Chen et al. [27] were, to our knowledge, the first to exploit window sparsity in local-attention vision transformers. They employed a coarse-grained pruning strategy that removes entire windows in Swin Transformer. A practical challenge is that, although attention is computed window-wise, the subsequent MLP layers operate globally as standard fully connected layers without window structure. To resolve this mismatch, they modified the MLP and layer-normalization operations to also operate in a window-wise manner. Window importance is measured simply using L2 activation magnitudes.

Kong et al. [35] propose a fine-grained token pruning framework, motivated by the observation that tokens exhibit varying redundancy across attention heads. They first estimate token importance for each head, then aggregate these per-head scores with learned head weights to obtain a global token importance score. Rather than discarding tokens that are considered as redundant, they merge low-importance tokens into a single package token to reduce information loss.

Although these methods can achieve higher sparsity, fine-grained token pruning often leads to irregular computation patterns that are difficult for existing hardware to exploit efficiently, making it challenging to realize theoretical latency improvements in practice. Coarse-grained, window-level pruning offers a more hardware-friendly alternative. However, Chen et al.'s [27] window pruning approach assumes that some windows are entirely redundant. In practice, this assumption rarely holds. Even visually uninformative background regions can carry useful structural information. Moreover, coarse-grained window removal can become overly aggressive when layers contain very few windows. For instance, in the base configuration of Swin Transformer, more than 90% of the layers contain four or fewer windows, leaving little granularity for safe pruning.

These observations suggest that a more balanced approach is needed that maintains the hardware efficiency of window-level operations while allowing selective pruning within each window. Motivated by this, in the following we present our window-level head pruning approach, which enables selective pruning within each window without disrupting window-level computation.

III. Background

A. Swin Transformer Overview

The Swin Transformer improves upon the original ViT design [14] by replacing global self-attention with window-based self-attention, enabling the model to scale linearly with increasing image resolution. Equation 1 shows the original attention calculation in ViT where $Q, K, V \in \mathbb{R}^{B\times H\times N\times C}$. Here, B, H, N, and C are the batch size, the number of heads, the number of patches and the head size, respectively.

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{1}$$

In window-based attention, the feature map is partitioned into W non-overlapping windows, each containing N/W patches. Attention is then computed independently within each window by reshaping the tensors such that $\hat{Q}, \hat{K}, \hat{V} \in \mathbb{R}^{\hat{B}\times H\times \hat{N}\times C}$, where $\hat{B} = B \cdot W$ and $\hat{N} = N/W$. This effectively treats each window as an independent token sequence.

A limitation of fixed windows is that tokens in neighboring windows cannot attend to one another, even when their content is highly correlated. Swin Transformer addresses this by applying a cyclic shift to the window partitioning in alternating layers, enabling cross-window information flow without employing global attention. Additionally, Swin employs a hierarchical design with patch merging, which gradually reduces spatial resolution while increasing channel dimensionality. As a result, deeper layers contain very few windows.

B. Multi-Head Self-Attention

The multi-head attention mechanism decomposes attention into H parallel heads, each learning distinct patterns or relationships:

$$\text{MHSA}(X) = \text{Concat}(h_1, \ldots, h_H)W_O, \tag{2}$$

where

$$h_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \tag{3}$$

The output projection W_O mixes information across heads. Each head operates over the same window, but captures different types of dependencies (e.g., local structure, long-range correlation, texture orientation). In principle, this diversity enables richer representations.

C. Head Redundancy in Attention

Despite the intended specialization of attention heads, numerous studies have shown that many heads contribute little to overall model performance [28], [29]. Some layers can be reduced to a single head, and moderate pruning may even improve accuracy due to reduced overfitting or noise suppression. These findings suggest that head importance is not uniform and that head redundancy is common in Transformer architectures.

In vision transformers, similar redundancy is very likely, particularly because window-based attention groups tokens according to spatial locality. Windows dominated by uninformative regions (e.g., background) may require fewer heads, whereas semantically rich windows (e.g., object boundaries) may benefit from multiple heads. This indicates that head relevance may vary *across windows*, not just across layers.

D. Computational Cost of Attention

The computational complexity of self-attention within a window of size $\hat{N}$ is:

$$\mathcal{O}(H\hat{N}^2C), \tag{4}$$

dominated by the QK^T matrix multiplication. The cost scales linearly with the number of heads H and quadratically with window size $\hat{N}$. Since Swin Transformer uses the same number of heads for every window, the runtime and FLOPs are uniformly allocated, even though different windows may require different representational capacity. Consequently, reducing the number of active heads in less informative windows has the potential to substantially reduce the computational cost without sacrificing accuracy.

E. Motivation for Window-Level Head Pruning

Although attention is computed independently within each window, all windows share the same number of heads. This uniform allocation overlooks spatial variation in token complexity. We hypothesize that attention head relevance is *window-dependent*, i.e., some windows contain rich semantic features that require multiple heads, while others are sufficiently represented with only a subset of heads. This motivates a pruning strategy that preserves the hardware-favoring window structure while selectively reducing heads within individual windows, leading directly to our proposed window-level head pruning approach.

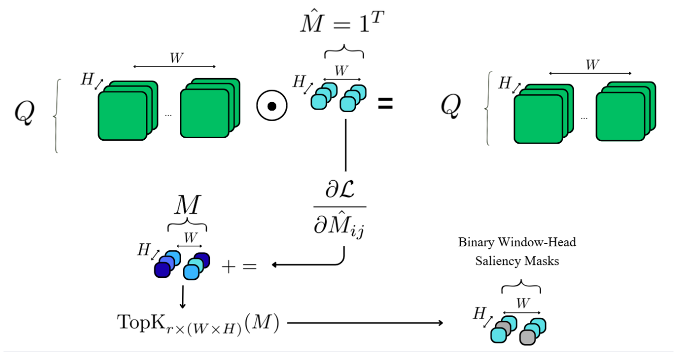
$\hat{M} = 1^T$
W
H
Q
$\odot$
=
Q
$\frac{\partial \mathcal{L}}{\partial \hat{M}_{ij}}$
M
$+=$
Binary Window-Head Saliency Masks
$\mathrm{TopK}_{r\times(W\times H)}(M)$

Fig. 2. Overview of the training pipeline used to learn window-level head saliency. Given an input image, we compute gradients with respect to the 1^T tensor $\hat{M}$, aggregate saliency scores, and iteratively update the pruning masks. In the mask, grey heads indicate pruned heads, while blue heads correspond to the remaining heads.

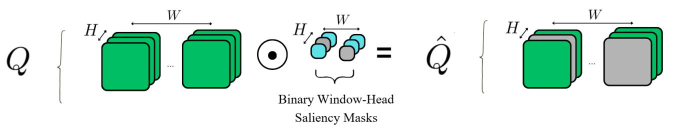

Fig. 3. Overview of the inference process, where the pruning masks learned during training are applied to remove redundant window–head pairs and reduce the computational cost of attention. Likewise, grey heads in the Q matrix show the pruned heads while the green ones are the remaining ones.

IV. Methodology

A. Determining Head Importance

Most existing pruning approaches for Transformers [27], [31] rely on magnitude-based criteria. Although intuitive, magnitude alone is a poor proxy for importance; the assumption that small weights contribute little to performance has been questioned extensively [36]. In contrast, gradient-based methods provide a more principled measure of saliency by capturing the sensitivity of the loss to perturbations of individual weights. A larger gradient magnitude indicates that modifying the corresponding parameter produces a larger change in the loss, and thus reflects its true contribution to the optimization landscape.

B. Global Pruning Rate and Layer-Wise Allocation

Unlike prior work, we do not specify fixed layer-wise pruning ratios. Instead, we define a *global pruning ratio* that determines the total number of window-head pairs to be pruned across the entire network. This design choice is motivated by two challenges. First, it is difficult to predict layer-wise importance a priori. A layer may appear critical overall while being dominated by only a few important window-head pairs, leaving the remainder safe to prune. Second, brute-force search over layer-level ratios is infeasible, since even the smallest Swin Transformer contains 16 layers, leading to a prohibitively large combinatorial space.

To address this, we compute gradients for all window-head pairs across all layers in a single backward pass. This yields a global set of saliency scores that implicitly allocates pruning ratios per layer based on actual gradient magnitudes rather than heuristic assumptions.

C. Gradient-Based Saliency Estimation

Our pruning strategy proceeds iteratively, as described in Algorithm 1. Each iteration begins by sampling two subsets of the training dataset: one for saliency estimation and another for fine-tuning after pruning. To estimate head importance, we multiply the query (Q), key (K), and value (V) tensors by a learnable binary gating tensor of shape $W \times H$, where W is the number of windows and H the number of heads. During the saliency backward pass, this gating tensor is set to an all-ones mask so that the gradients reflect the contribution of each window-head pair. We average the gradients across all samples in the saliency batch to obtain a stable estimate.

Let $g_{w,h}$ denote the gradient magnitude associated with window w and head h. To prevent re-pruning during subsequent iterations, previously removed pairs are assigned a saliency of $+\infty$. We sort all saliencies in ascending order and select the i-th smallest value as the threshold t, guaranteeing that exactly i currently active pairs fall below the threshold. A binary pruning mask is then constructed by marking all window-head pairs with saliency less than t. These pairs are excluded from the computation for the current iteration.

Figures 2 and 3 illustrate how the masks are generated during training and applied during inference. For the sake of illustration, the masks are shown as being applied after the $Q/K/V$ projections. In practice, this is inefficient because it prevents any computation savings during $Q/K/V$ generation. In our implementation, we instead decompose the projection weights into per-head blocks before the attention operation and selectively apply these head-specific weights to each window according to the saliency mask.

D. Iterative Pruning and Fine-Tuning

After each pruning step, the model is fine-tuned using the small training subset $\hat{D}$ for one epoch to recover from the structural change. Once the global pruning ratio is reached, the model undergoes a final fine-tuning stage on the full training dataset for 15 epochs. This two-stage fine-tuning scheme significantly stabilizes training and mitigates accuracy degradation caused by pruning.

V. Experimental Setup

A. Vision Transformer Models

We evaluated our approach on two configurations of the Swin Transformer: Swin-Tiny and Swin-Base. These models differ primarily in two aspects. First, their embedding dimensions are 96 and 128 for Swin-Tiny and Swin-Base, respectively. Second, although both models share the same number of layers in stages 1, 2, and 4, they differ substantially in stage 3, which contains 6 layers in Swin-Tiny and 18 layers in Swin-Base.

We selected these two variants to assess the effectiveness of our method across architectures of varying capacity and computational complexity. For both models, we do not prune the

Algorithm 1 Pruning Algorithm

```
Input: Training dataset $\mathcal{D}$, initial model $\mathcal{V}$ with $L$ layers, global pruning ratio $r$, number of pairs pruned per iteration $i$.
Output: The pruned Swin Transformer.
1: Initialize mask $M \leftarrow \mathbf{1}_L$; current pruning ratio $r_{\text{cur}} \leftarrow 0$.
2: while $r_{\text{cur}} < r$ do
3:     Randomly sample a large subset $\hat{\mathcal{D}} \subset \mathcal{D}$ for training;
4:     Randomly sample a small subset $\hat{d} \subset \mathcal{D}$ for saliency estimation;
5:     Compute window–head gradients $G$ using $\hat{d}$;
6:     for $w = 0$ to $W - 1$ do
7:         for $h = 0$ to $H - 1$ do
8:             $g[w,h] \leftarrow \frac{1}{|\hat{d}|} \sum_{x \in \hat{d}} G[x,w,h]$;
9:         end for
10:    end for
11:    $\mathcal{P} \leftarrow \{(w,h) \mid M[w,h] = 0\}$;
12:    $g[w,h] \leftarrow +\infty$ for all $(w,h) \in \mathcal{P}$; // avoid re-pruning
13:    $(\text{indices}, \text{vals}) \leftarrow \text{Sort}(g)$;
14:    $t \leftarrow \text{vals}[i]$;                    // pruning threshold
15:    $\text{mask}[w,h] \leftarrow (g[w,h] < t)$;
16:    $M[w,h] \leftarrow 0$ for all $(w,h)$ with $\text{mask}[w,h] = 1$;
17:    Fine-tune $\mathcal{V}$ on $\hat{\mathcal{D}}$ for one epoch;
18:    $r_{\text{cur}} \leftarrow r_{\text{cur}} + i$;
19: end while
20: Fine-tune $\mathcal{V}$ on $\mathcal{D}$ for 15 epochs;
```

layers in the final stage, as these layers contain only a single window, making window-level head pruning inapplicable.

B. Training Setup

All experiments were conducted on the ImageNet-1K classification dataset [37], using the standard training split for optimization and the validation split for evaluation. During iterative pruning and fine-tuning, we apply knowledge distillation [24], where the teacher model is the corresponding unpruned counterpart. The total loss is a weighted combination of the cross-entropy loss and the distillation loss. We set the distillation weight to $\alpha = 0.7$ and the temperature to $T = 2$. Optimization is performed using AdamW [38] with a learning rate of 5×10^{-5} and a weight decay of 1×10^{-4}.

For the final fine-tuning stage, we maintain the same base learning rate and weight decay, but introduce a cosine learning rate schedule with a 10% warm-up period. The learning rate increases linearly from 1×10^{-6} to the base value during warm-up, and then follows a cosine decay back to 1×10^{-6} for the remainder of training. This schedule is implemented using the `CosineLRScheduler` [39].

VI. Results and Discussion

We first report the classification accuracies achieved by the two model configurations under different global pruning ratios r, shown in Figures 4 and 5. For both Swin-Tiny and Swin-Base, we evaluate four uniformly spaced pruning ratios between $r \approx 0.33$ and $r \approx 0.67$ (inclusive). Due to divisibility

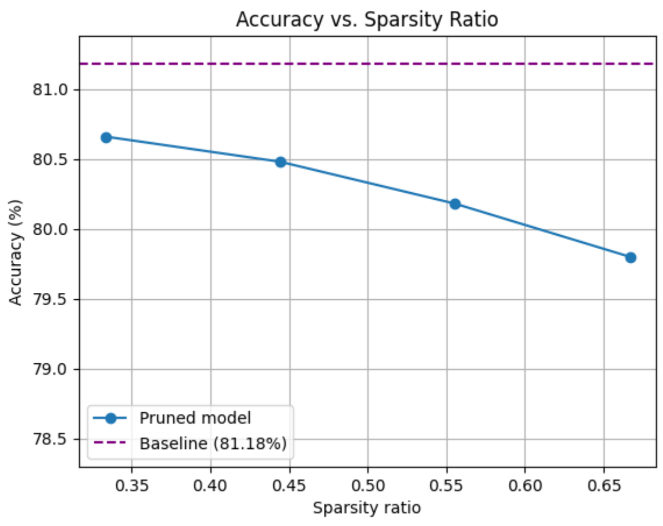

Fig. 4. Swin-Tiny Classification Accuracies

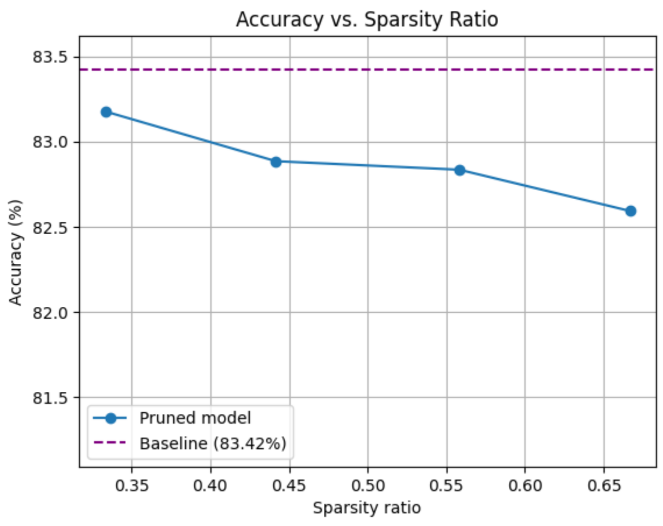

Fig. 5. Swin-Base Classification Accuracies

constraints, the exact pruning ratios differ slightly between the two configurations (except at the endpoints), but are matched as closely as possible to ensure fair comparison.

Overall, Swin-Base exhibits slightly higher robustness to pruning, likely due to its larger representational capacity. Nevertheless, Swin-Tiny also maintains competitive accuracy, with only a $\sim 1.5\%$ drop at the highest pruning level $r \approx 0.67$. At smaller pruning ratios, such as $r = 0.33$, both configurations retain most of their accuracy, with drops of only 0.5% for Swin-Tiny and 0.2% for Swin-Base.

Figures 6 and 7 visualize the resulting layer-wise pruning ratios for Swin-Tiny under various global pruning ratios. At $r \approx 0.67$, the pruning ratio generally decreases with network depth. One might be tempted to conclude that a monotonically decreasing, depth-dependent pruning schedule would suffice. However, the pattern at $r \approx 0.33$ deviates substantially from

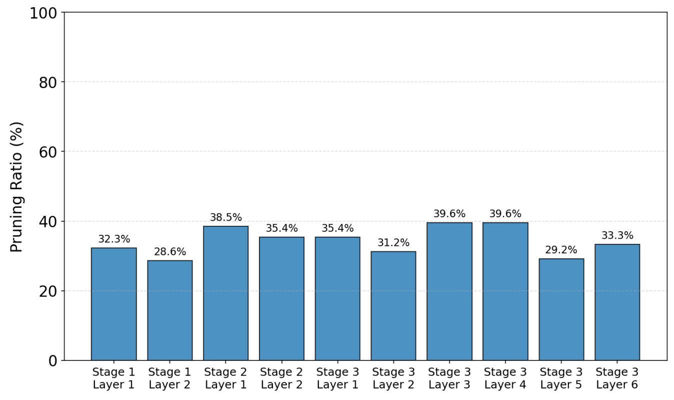

Fig. 6. Layer-wise Pruning Ratios ($r \approx 0.33$)

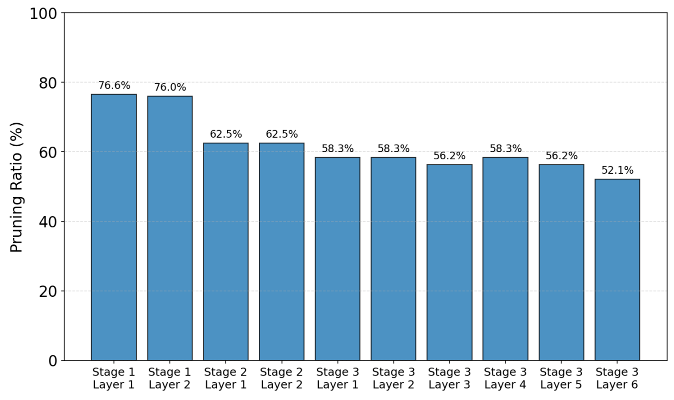

Fig. 7. Layer-wise Pruning Ratios ($r \approx 0.67$)

this behavior: the layer-wise pruning ratios are markedly more irregular. This discrepancy underscores the difficulty of designing hand-crafted, layer-wise pruning rules, and highlights the need for data-driven allocation of pruning across layers.

Table I reports the MAC reductions for the entire attention computation, including the query/key/value projections, the self-attention computation, and the output projection. Because our pruning mask is learned offline, no additional inference overhead is introduced, unlike methods that require prediction modules or dynamic mask generation. The total MAC reduction does not exactly match r because the final stage of Swin, which contains a single window, is not pruned. Swin-Base achieves larger relative MAC reductions at equivalent pruning ratios, owing to its third stage containing 18 layers, which dominates its total computational cost.

We additionally evaluate latency to assess the practicality of our approach for real-time applications. Note that these measurements do not correspond to end-to-end latency, as a full evaluation would require an optimized implementation that efficiently maps the irregular, pruned computations with varying window sizes onto the GPU; we leave this system-level optimization to future work. Instead, we measure the attention latency separately for each window and then aggregate the per-window latency over the network for both the

TABLE I
MAC REDUCTION UNDER DIFFERENT PRUNING RATIOS.

r	Swin-Base		Swin-Tiny	
(pruning ratio)	# of MACs	Red. (%)	# of MACs	Red. (%)
unpruned	5.24×10^9	–	2.61×10^9	–
0.33	3.63×10^9	30.7	1.88×10^9	28.0
0.44	3.10×10^9	40.9	1.63×10^9	37.3
0.55	2.56×10^9	51.1	1.39×10^9	46.6
0.67	2.03×10^9	61.3	1.15×10^9	56.0

pruned and unpruned models to enable a fair comparison. On Swin-Base, we observe latency reductions ranging from 22.43% at $r \approx 0.33$ to 51.39 % at $r \approx 0.66$, demonstrating the potential of our method to substantially reduce attention latency in latency-sensitive deployments.

The results demonstrate that window-level head pruning is an effective strategy for reducing attention computation while preserving accuracy across differently sized Swin Transformer configurations. Notably, the learned pruning patterns reveal that redundancy is not uniformly distributed across layers, nor does it follow simple depth-dependent trends. This validates our choice of a global, gradient-driven saliency measure, rather than manually prespecified layer-wise pruning ratios. Furthermore, the substantial MAC reductions achieved, particularly in Swin-Base configuration, indicate that our method scales favorably with model depth and complexity. Importantly, because the pruning mask is static and requires no runtime prediction, the approach retains the hardware efficiency of window-level attention and avoids irregular computation patterns associated with fine-grained token pruning. These findings suggest that selective head pruning within windows offers a promising direction for designing efficient vision transformers that maintain accuracy while substantially reducing computational overhead.

VII. Conclusion

In this work, we introduced SlimSwin, a gradient-based window-level head pruning method for Swin Transformer, founded on the observation that different windows require different amounts of attention capacity. By applying a learnable mask tensor over window-head pairs and using gradients of this mask as saliency scores, our approach applies pruning globally across all layers without relying on manually specified layer-wise pruning ratios. The resulting pruning masks are static and window-aligned, preserving the hardware-friendly structure of window-based attention while selectively removing redundant computation within each window.

In future work, we aim to extend this method to dynamically determine pruning masks during inference, with the potential to further improve accuracy. In addition, we plan to complement our attention-level reductions by applying pruning to the MLP layers, enabling additional computational savings.

References

[1] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[11] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/3416a75f4cea9109507cacd8e2f2aefc-Paper.pdf

[12] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen, "Axial-deeplab: Stand-alone axial-attention for panoptic segmentation," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 108–126.

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[14] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[15] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[16] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[17] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989.

[18] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," *Advances in neural information processing systems*, vol. 5, 1992.

[19] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[20] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[21] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.

[22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[23] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[25] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," *Advances in neural information processing systems*, vol. 34, pp. 13 937–13 949, 2021.

[26] Y. Tang, K. Han, Y. Wang, C. Xu, J. Guo, C. Xu, and D. Tao, "Patch slimming for efficient vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 165–12 174.

[27] X. Chen, Z. Liu, H. Tang, L. Yi, H. Zhao, and S. Han, "Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2061–2070.

[28] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" *Advances in neural information processing systems*, vol. 32, 2019.

[29] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.

[30] Y. Ding, Y. Liu, H. Luan, and M. Sun, "Visualizing and understanding neural machine translation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1150–1159. [Online]. Available: https://aclanthology.org/P17-1106/

[31] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 97–110.

[32] P. Dupuis and H. Ishii, "On lipschitz continuity of the solution mapping to the skorokhod problem, with applications," *Stochastics: An International Journal of Probability and Stochastic Processes*, vol. 35, no. 1, pp. 31–62, 1991.

[33] K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural networks*, vol. 6, no. 6, pp. 801–806, 1993.

[34] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[35] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang *et al.*, "Spvit: Enabling faster vision transformers via latency-aware soft token pruning," in *European conference on computer vision*. Springer, 2022, pp. 620–640.

[36] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," *arXiv preprint arXiv:1802.00124*, 2018.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[39] ——, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.