Universität Stuttgart

Institute for Artificial Intelligence
Machine Learning for Simulation Science

Universitätsstraße 32
70569 Stuttgart

Master Thesis

Accelerating Segment Anything Models via Token Merging: A Comparative Study and a Spectrum Preservation-Based Approach

Siwei Xie

Study program:	M.Sc Informatik
1. Examiner:	Prof. Dr. Mathias Niepert
2. Examiner:	Prof. Dr. Steffen Staab
Advisor:	M.Sc. Duy Nguyen
start date:	2025-01-02
end date:	2025-04-29

Abstract

The Segment Anything Model (SAM) has emerged as a significant advancement in image segmentation, demonstrating exceptional generalization across diverse datasets with minimal task-specific tuning. However, its computational demands, inherited from Vision Transformers (ViTs), pose considerable challenges for deployment in resource-constrained environments. This thesis addresses these challenges by integrating token merging strategies, which have proven effective in enhancing the efficiency of ViTs without additional training. Specifically, we conduct a comprehensive analysis of SAM's architecture and adapt existing token merging techniques to reduce computational overhead while maintaining high segmentation accuracy. We propose an architecture for SAM that incorporates these strategies and evaluate its performance and computational efficiency across various datasets, showing that our approach effectively accelerates SAM's inference speed while preserving segmentation quality. Furthermore, we propose *GradToMe* based on *PiToMe*, an innovative method that leverages gradient approximation and grid-based sampling to combine similar tokens. This approach emphasizes spectrum preservation to retain critical information during the token reduction process, thereby improving the effectiveness of token merging and further saving computational costs. Consequently, our results demonstrate that this approach enhances the feasibility of deploying SAM in real-time applications, making it more suitable for use in resource-limited environments without compromising performance. Code is available at: https://github.com/xxjsw/tome_sam.

Kurzfassung

Das Segment Anything Model (SAM) hat sich als ein bedeutender Fortschritt in der Bildsegmentierung etabliert und zeigt außergewöhnliche Generalisierungsfähigkeiten über verschiedene Datensätze hinweg, bei minimaler aufgabenspezifischer Feinabstimmung. Allerdings stellen die hohen rechnerischen Anforderungen, die vom Vision Transformer (ViT) übernommen wurden, erhebliche Herausforderungen für den Einsatz in ressourcenbeschränkten Umgebungen dar. Diese Thesis geht diese Herausforderungen an, indem sie Token-Merging-Strategien integriert, die sich als effektiv erwiesen haben, um die Effizienz von ViTs ohne zusätzliche Trainingsphase zu verbessern. Insbesondere führen wir eine umfassende Analyse der SAM-Architektur durch und passen bestehende Token-Merging-Techniken an, um den Rechenaufwand zu verringern, ohne die Segmentierungsgenauigkeit zu beeinträchtigen. Wir schlagen eine Architektur für SAM vor, die diese Strategien integriert, und evaluieren ihre Leistung sowie ihre rechnerische Effizienz über verschiedene Datensätze hinweg. Dabei zeigen wir, dass unser Ansatz die Inferenzgeschwindigkeit von SAM effektiv beschleunigt, während die Segmentierungsgenauigkeit erhalten bleibt. Darüber hinaus schlagen wir mit *GradToMe*, basierend auf *PiToMe*, eine innovative Methode vor, die Gradientenapproximation und gitterbasierte Stichproben nutzt, um ähnliche Tokens zu identifizieren. Diese Methode legt besonderen Wert auf die Erhaltung des Spektrums, um sicherzustellen, dass während des Token-Merging-Prozesses kritische Informationen erhalten bleiben, was den Token-Merging-Prozess optimiert und die Inferenzgeschwindigkeit weiter steigert. Unsere Ergebnisse zeigen, dass dieser Ansatz die Machbarkeit des Einsatzes von SAM in Echtzeitanwendungen verbessert, wodurch es besser für den Einsatz in ressourcenbegrenzten Umgebungen geeignet ist, ohne die Leistung zu beeinträchtigen. Der Code ist verfügbar unter folgendem Link: https://github.com/xxjsw/tome_sam.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, Vision Transformers (ViTs) [18] have reshaped the landscape of computer vision by delivering exceptional performance across a range of tasks, from image classification [10, 18, 40, 48, 74, 97] to semantic segmentation [14, 71, 79, 82, 100] and even image generation [19, 56, 94]. Their core strength lies in the self-attention [76] mechanism, which enables the modeling of long-range dependencies and rich global context. However, this mechanism comes at a steep cost: the computational complexity of self-attention grows quadratically with the number of input tokens. The need to process large numbers of tokens limits their usability in scenarios with constrained hardware resources or latency-sensitive applications. This poses severe challenges for efficiency and scalability when dealing with high-resolution images, which are common in practical applications. The trade-off between accuracy and efficiency is particularly pronounced in dense prediction tasks such as semantic segmentation, where maintaining spatial detail and per-pixel prediction fidelity is crucial.

To address this issue, an increasing number of studies [3, 5, 6, 8, 9, 11, 20, 24, 30, 31, 33, 39, 42, 43, 50, 52, 53, 54, 55, 60, 62, 65, 66, 68, 72, 75, 78, 81, 102] have explored token reduction strategies, aimed at minimizing the computational burden without sacrificing output quality. These strategies seek to reduce the number of tokens processed by the layered ViT blocks, ultimately lowering the required computational steps and enabling faster training and inference. Among these studies, Token Merging (ToMe) [3] has emerged as one of the most innovative and pioneering techniques. Token merging works by identifying and combining similar tokens, thereby minimizing unnecessary computational redundancy. Without merging, many duplicated tokens are processed independently, resulting in inefficient use of computational resources.

Unlike token pruning methods [20, 33, 43, 53, 60, 72, 78], which simply discard tokens, risking the complete loss of valuable information crucial for accurate predictions, token merging [3, 5, 11, 30, 31, 54, 68, 75] preserves the semantic structure of the input. It intelligently groups tokens that represent similar features or regions of interest, resulting in a more compact and informative token representation. Token pruning is widely applied in image classification, where global representations are typically sufficient. However, in segmentation tasks, every pixel—or, in transformer-based models, every token—should contribute to the final prediction and thus requires individual consideration. Token merging is particularly advantageous in such cases, as it facilitates the preservation of spatial detail essential for accurate results. Furthermore, when token merging is applied, "unmerging" offers a more intuitive process for recovering the original input resolution from the merged tokens. In contrast, pruning results in complete information loss, making it difficult to associate remaining tokens with discarded ones, and therefore less compatible with dense pixel tasks. Moreover, advanced token merging algorithms such as *PiToMe* [75] preserve critical spatial and semantic relationships. This capability is essential for segmentation tasks, where fine-grained distinctions between neighboring regions are often required. By reducing the token count with careful considerations, token merging methods effectively compress the token space, alleviating the computational burden without significantly compromising model performance. Consequently, token

merging constitutes a more reliable alternative to pruning, especially in contexts demanding both computational efficiency and high-precision output, such as real-time segmentation or deployment on resource-constrained edge devices.

One of the most influential developments in segmentation has been the Segment Anything Model (SAM) [32], a foundation model capable of performing high-quality segmentation across diverse datasets with minimal task-specific finetuning. Despite its impressive performance and generalization capabilities, SAM inherits the computational intensity of ViTs, making its deployment in resource-limited environments a considerable challenge. Accordingly, we adapt recently proposed bipartite soft matching (BSM)-based token merging strategies [3, 5, 75], which were originally developed to enhance the efficiency of vision transformers without requiring additional training and designed for efficient parallel execution, to the SAM. By integrating these methods, our goal is to reduce the computational complexity and enhance the throughput of SAM, while preserving segmentation accuracy, thereby improving its suitability for broad deployment.

The main contributions of this thesis are:

- We conduct a comprehensive analysis of the Segment Anything Model (SAM) architecture and investigate the adaptation of existing token merging strategies to enhance its computational efficiency.
- Building upon a detailed understanding of SAM's architecture, we integrate token merging techniques and systematically evaluate their impact on performance and computational efficiency across multiple extremely fine-grained datasets. Furthermore, we perform an in-depth analysis of the experimental results, uncovering additional insights and providing supporting evidence.
- We analyze the limitations of existing token merging methods and introduce *GradToMe*, an approach based on first-order gradient approximation and grid-based sampling to capture similar tokens. This method aims to enhance the effectiveness of the token merging process through spectrum preservation and further reasonably decrease computational overhead.

2 Related Work

Efficient Vision Transformers Vision Transformers (ViTs) have revolutionized computer vision by leveraging the self-attention mechanism to capture global context, but face significant computational and memory challenges due to the inherent complexity of the self-attention mechanism, particularly in high-resolution and real-time applications. These limitations present substantial barriers to deployment on resource-constrained devices, and achieving similar results with a more efficient architecture is not straightforward. In response, an expanding body of research has focused on designing more efficient ViT architectures that maintain competitive performance while alleviating resource demands. One major research direction involves architectural modifications aimed at reducing the cost of self-attention, while preserving the global contextual understanding that is central to ViT models. For example, EfficientViT [46] incorporates a subsampling layer along with cascaded group attention; Swin Transformer [48] introduces a shifted window attention mechanism; and PVT [79] employs a pyramid structure with spatial-reduction attention (SRA), progressively decreasing output resolution over four stages. Similarly, PoolFormer [91] replaces global token mixing with a simple pooling operator for basic token aggregation. Moreover, several works directly tackle the quadratic complexity of self-attention, aiming to achieve sub-quadratic or even linear computational scaling. Approaches such as SOFT [51] and SimA [35] propose softmax-free operation for the attention mechanism. Hydra Attention [4] introduces a linear attention mechanism based on decomposable kernel strategies, and Castling-ViT [88] proposes a fully linear architecture through a linear-angular attention module. A third direction explores parameter pruning, where unimportant parameters and their associated computations (e.g., by removing attention heads) are discarded to enhance efficiency. Representative methods include WDPruning [89], NViT [86], and X-Pruner [90], which design specialized pruning frameworks to reduce overhead. In addition, knowledge distillation [25] has been employed to train lightweight student models by transferring knowledge from deeper, more complex teacher models. Notable models include DeiT [74], TinyViT [80], CivT [61] and MiniViT [96]. Furthermore, quantization techniques [27] are leveraged in studies [17, 41, 47, 92] to reduce memory and computational requirements by lowering the numerical precision of model parameters and activations. Lastly and importantly, token reduction strategies aim to reduce the number of image patches processed by the model without modifying the underlying architecture or the self-attention mechanism directly. This approach constitutes the core motivation of our work on accelerating the Segment Anything Model (SAM), where we apply token reduction techniques to improve efficiency without compromising model performance.

Token Reduction in general While Vision Transformers (ViTs) suffer from quadratic complexity in self-attention, their efficient matrix operations and minimal assumptions in inputs make them highly adaptable and hardware-friendly. These strengths have driven interest in token reduction methods that improve efficiency without altering the core model design. One prominent token reduction strategy is *token pruning*, which has been widely explored in efficient ViT research. These

methods aim to identify and eliminate less informative tokens using various criteria. For example, some approaches [33, 53, 60, 72] employ auxiliary heads to detect unimportant tokens, while others [20, 43] leverage attention weights to retain only the most influential tokens. Additionally, two-stage approaches [78] combine importance scores and token similarity to guide pruning decisions. The tokens predicted as less informative can then either be completely discarded [20, 53, 60, 72] or fused into a single representative token [33, 43, 78]. Another line of work is *token halting*, which reduces computational cost by freezing the processing of tokens deemed less important for certain layers. Token halting decisions can be guided by auxiliary heads [73] or by halting scores derived from existing model parameters, without introducing additional parameters [87]. A third, and more relevant strategy to our study, is *token merging*, which aims to compress the token set by combining similar or redundant tokens. Certain approaches introduce learnable layers that map the original tokens into a smaller, more representative set [55, 62, 65, 66, 102]. Meanwhile, a few studies adopt clustering-based techniques [9, 39, 42, 52] to iteratively group and merge similar tokens. However, clustering methods are typically not parallelizable and are thus inefficient for batch processing. An alternative and more efficient solution is bipartite soft matching (BSM), which greedily selects token pairs with the highest similarity scores and merges them. This method, initially introduced in *ToMe* [3], enables parallel execution and has been extended in several follow-up works [5, 11, 24, 30, 31, 54, 68, 75] across various domains, including image classification, semantic segmentation, and diffusion models. In addition, several studies combine the aforementioned pruning, halting, fusing or merging techniques to achieve effective token reduction [6, 8, 11, 50, 81].

Token Reduction for Segmentation For dense pixel tasks such as segmentation, accurately recovering spatial resolution after token reduction is crucial. In the case of *token pruning*, associating discarded tokens with retained ones is non-trivial, as pruned tokens are deemed less informative and typically have minimal influence on final predictions, making this method more suitable for classification tasks. However, in segmentation, every pixel requires a prediction, regardless of its relative importance. *Token halting*, while maintaining the original resolution by merely skipping certain tokens in specific layers rather than removing them, risks information loss due to incomplete self-attention computation. In contrast, *token merging* addresses these limitations by consolidating highly similar tokens, offering a more balanced trade-off between efficiency and performance in dense prediction tasks.

Among existing token merging approaches, clustering-based methods require iterative computation and cannot be vectorized for parallel batch processing. While they reduce the token sequence and thus enhance Transformer efficiency, the clustering process itself is computationally inefficient. Similarly, methods that rely on additional learned layers to map the full token set to a smaller one introduce extra parameters and require training, limiting their transferability to new ViT backbones. In contrast, bipartite soft matching (BSM) methods proposed by *ToMe* [3] are non-iterative, training-free, and easily integrable into various ViT architectures, making them a preferred choice in our work. This merging process is guided by a lightweight matching algorithm that identifies and combines redundant tokens, accelerating both training and inference. This approach achieves speed improvements comparable to token pruning but maintains higher accuracy by avoiding the information loss associated with outright token removal.

Several studies have successfully applied BSM-based token merging to ViT backbones for segmentation tasks, achieving efficiency gains while maintaining segmentation accuracy. ALGM [54] introduces a two-stage merging strategy: it performs local merging within a 2×2 window in the initial layer, followed by global merging in a deeper layer. Both stages adaptively merge only the most similar tokens exceeding predefined similarity thresholds. Evaluated on plain ViT backbones such as Segmenter [71] and SETR [100], ALGM not only improves efficiency but also enhances segmentation performance after training. Despite its merits, this approach is specifically designed for standard ViT architectures, and its adaptability to modified ViT variants that integrate other architectural enhancements remains uncertain. Most relevant to our work is Segformer++ [30], which integrates ToMeSD [5] and 2D neighbour merging into the convolution-assisted Segformer backbone [82], which is designed with a four-stage multi-scale output structure and incorporates spatial reduction attention. This work explores various merging configurations, showing that aggressive merging yields significant speedups at the cost of noticeable accuracy degradation, while moderate merging offers a more balanced trade-off between speed and performance.

Existing BSM-based token merging approaches have primarily been developed for pure Vision Transformer backbones and are not directly applicable to more complex architectures, such as the Segment Anything Model (SAM). SAM incorporates architectural modifications that complicate the direct integration of standard token reduction techniques. To address these challenges, we adapt and extend existing token merging strategies, tailoring them specifically to leverage SAM's structural characteristics and achieve efficient token reduction while preserving segmentation performance. Furthermore, based on our analysis of experimental results from integrating various token merging algorithms into SAM under different configurations, we propose another approach that builds upon *PiToMe* and incorporates first-order gradient approximation by Sobel operator, aiming to overcome current limitations and broaden the solution space available for token reduction.

3 Preliminaries

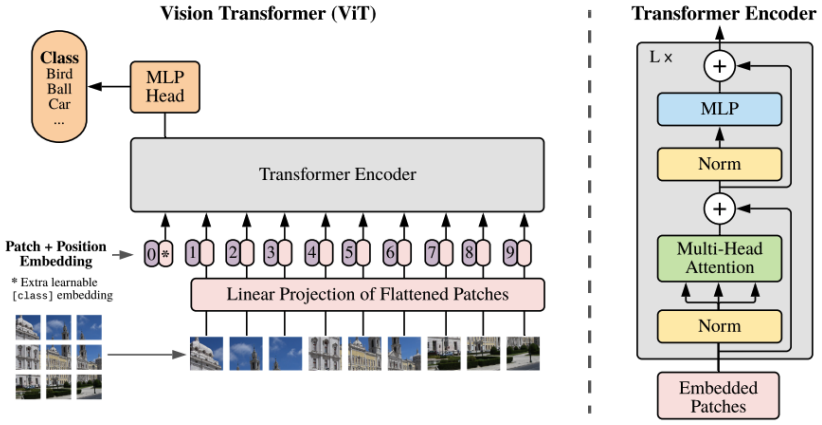
3.1 Vision Transformer & Self-Attention Mechanism

Transformers, introduced by Vaswani et al. in "Attention is All You Need" [76], revolutionized natural language processing (NLP) by utilizing self-attention mechanisms to capture dependencies across an entire sequence, allowing for parallel processing and efficient learning of long-range relationships. This contrasts with earlier models like recurrent neural networks (RNN) and long short-term memory networks (LSTM) [26], which process data sequentially. Models such as BERT [16] and GPT [59] have set new benchmarks in various NLP tasks, demonstrating the transformer architecture's superiority in tasks requiring contextual understanding over long distances in text.

The success of transformers in NLP led to their adoption in computer vision, culminating in the Vision Transformer (ViT) [18]. CNN [36] and ResNet [23] were the most successful models in the vision area, unlike these convolution-based models, which focus on local feature extraction, ViTs treat images as sequences of fixed-size patches and apply self-attention mechanisms to capture global dependencies across the entire image. ViTs are highly effective in capturing long-range dependencies, especially when trained on large datasets, which contributes to their success of outperforming traditional models in tasks requiring global context and detailed understanding. The Segment Anything Model (SAM), which is a Vision Transformer-based foundation model, leverages these strengths for promptable segmentation tasks.

3.1.1 Vision Transformer (ViT)

Figure 3.1: Standard Vision Transformer [18].



The Vision Transformer processes an image by first dividing it into non-overlapping patches, each of which is flattened and linearly projected into a fixed-dimensional embedding. These patch embeddings, along with an optional learnable class token, are augmented with positional encoding to retain spatial information and then passed as a sequence to a standard Transformer encoder. The encoder consists of multiple Transformer blocks, where each block applies multi-head self-attention and feedforward layers to model global dependencies, enabling the final representation to capture meaningful features.

1. Input Representation

1. **Image Input:** Given an image $X \in \mathbb{R}^{H \times W \times C}$, where H and W are the image height and width, and C is the number of channels.
2. **Patch Partitioning:** The image is divided into N non-overlapping patches, each of size $P \times P$:
   - The total number of patches is $N = (H \times W)/P^2$
   - Each patch is flattened into a vector $x_i \in \mathbb{R}^{P^2C}$, forming the sequence: $X' = [x_1, x_2, ..., x_N] \in \mathbb{R}^{N \times (P^2C)}$
3. **Linear Projection:** Each patch embedding is mapped into a d-dimensional latent space using a trainable projection matrix $\mathbf{E} \in \mathbb{R}^{(P^2C) \times d}$: $z_i = x_i\mathbf{E}, \quad \forall i \in \{1, ..., N\}$

   This results in: $Z = [z_1, z_2, ..., z_N] \in \mathbb{R}^{N \times d}$
4. **(Optional) Class Token for Classification:** Originally it was designed for classification tasks, so a learnable class token $z_{\text{cls}} \in \mathbb{R}^d$ is prepended to the sequence and can be later used for classification: $Z = [z_{\text{cls}}, z_1, z_2, ..., z_N] \in \mathbb{R}^{(N+1) \times d}$
5. **Positional Encoding:** Since transformers lack inherent spatial awareness, a learnable *positional embedding* $E_{\text{pos}} \in \mathbb{R}^{N \times d}$ is added: $Z_0 = Z + E_{\text{pos}}$, and this tokenized representation Z_0 is then passed to the transformer encoder for further processing.

2. Transformer Encoder The sequence Z_0 is processed through L stacked Transformer encoder layers. Each encoder layer consists of: Multi-Head Self-Attention (**MHSA**), Layer Normalization **LN**), Feed-Forward Network (**FFN**) and Residual Connections.

A single Transformer encoder layer updates the representations by sequentially applying a **MHSA** block and a **FFN**, each followed by a residual connection, as follows:

$$Z'_l = \text{MHSA}(\text{LN}(Z_{l-1})) + Z_{l-1},$$
$$Z_l = \text{FFN}(\text{LN}(Z'_l)) + Z'_l,$$

where l denotes the layer index.

3.1.2 Self-Attention Mechanism

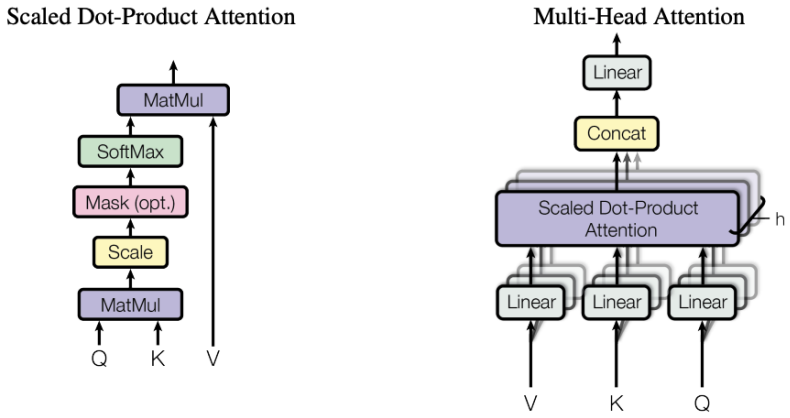


Figure 3.2: Self-Attention & Multi-head Self-Attention [76].

Self-attention is the core mechanism in Transformers that enables each patch in the image to capture global dependencies by attending to all other patches in the sequence. Given an input sequence $Z \in \mathbb{R}^{N \times d}$, self-attention is computed as follows:

1. **Linear Projections:** To obtain the *key*, *query* and *value* matrices, the input sequence undergoes three independent linear transformations:

   $$Q = ZW_Q, \quad K = ZW_K, \quad V = ZW_V$$

   where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$ are learnable weight matrices, and the resulting projections are: $Q, K, V \in \mathbb{R}^{N \times d_k}$.

2. **Scaled Dot-Product Attention:** The attention scores, which quantify the similarity between query and key representations, are computed as:

   $$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

   where the division by $\sqrt{d_k}$ stabilizes gradients by preventing excessively large values in the softmax function.

3. **Weighted Sum:** The attention output is obtained by multiplying the attention scores with the *value* matrix: $\text{Attn}(Z) = AV$.

4. **Final Output Projection:** The final output is transformed by a learnable projection matrix: $Z' = \text{Attn}(Z)W_O$, where $W_O \in \mathbb{R}^{d_k \times d}$ is a trainable weight matrix that projects the attended values back to the original embedding dimension d.

3.1.2.1 Multi-head Self-Attention

To enhance the expressiveness of self-attention, the single-head attention mechanism is extended to multiple heads, allowing the model to capture diverse features from different representation subspaces. Instead of computing a single set of *key*, *query* and *value* matrices, h independent attention heads are employed. Each head computes its own attention output as follows:

$$\text{head}_i = \text{Attn}(Z) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

where $Q_i = ZW_Q^i$, $K_i = ZW_K^i$, and $V_i = ZW_V^i$ are the projections for the i-th attention head, with learnable parameters $W_Q^i, W_K^i, W_V^i \in \mathbb{R}^{d \times d_k}$. The outputs of all heads are then concatenated and linearly projected: $\text{MHSA}(Z) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$, where $W_O \in \mathbb{R}^{hd_k \times d}$ is a learnable output projection.

3.1.3 Computational Complexities

We provide a theoretical analysis of the computational complexity for each component of the ViT encoder, expressed in Big-O notation, with respect to input size and required embedding dimensionality, as summarized in **Table 3.1**. In a Vision Transformer encoder, the dominant factor in computational cost is the self-attention mechanism in each transformer block, which exhibits quadratic complexity with respect to the number of patches (i.e., $O(N^2)$). This quadratic scaling of the self-attention operation becomes a significant limitation, particularly as the image resolution increases, leading to a rapid growth in computational demands.

In this work, we address the computational challenges inherited by SAM through the application of token merging techniques. By merging similar or redundant patches into a single representation, we reduce the total number of patches N, thereby decreasing the computational burden of the self-attention mechanism, and optionally, that of other modules. This reduction in patch count significantly enhances the efficiency of Vision Transformers, improving their scalability for high-resolution image processing and their feasibility for real-time applications.

Processing Step	Computational Complexity
Patch Embedding	$O(N \cdot P^2 \cdot C \cdot d)$
Positional Encoding	$O(N \cdot d)$
Self-Attention (MHSA)	Overall:$O(h \cdot N^2 \cdot d_k) = O(N^2 \cdot d)$
	1. Linear Projection (Q, K, V): $O(3 \cdot h \cdot N \cdot d \cdot d_k) = O(3 \cdot N \cdot d^2)$
	2. Attention Scores (QK^T computation): $O(h \cdot N^2 \cdot d_k) = O(N^2 \cdot d)$
	3. Softmax Operation: $O(h \cdot N^2)$
	4. Weighted Sum ($A \cdot V$): $O(h \cdot N^2 \cdot d_k) = O(N^2 \cdot d)$
	5. Final Projection: $O(N \cdot d^2)$
Feedforward Network (FFN)	$O(N \cdot d^2)$
Layer Normalization	$O(N \cdot d)$

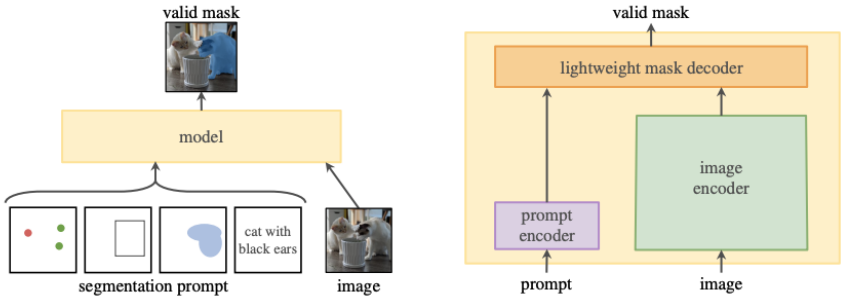
- N: Number of patches in the input sequence
- P: Patch size
- C: Number of input channels
- d_k: Dimension of each attention head
- h: Number of attention heads
- $d = h \cdot d_k$: Embedding dimension of each patch

Table 3.1: Computational Complexities of Vision Transformer Encoder and Self-Attention Mechanism.

3.2 Segment Anything(SAM)

Segment Anything Model (SAM) [32] is a pioneering vision foundation model, developed to handle a wide range of segmentation tasks through prompts. This approach allows SAM to generate segmentation masks based on various prompts, enabling zero-shot transfer to numerous segmentation challenges without additional training. Another major contribution of the SAM project is the SA-1B dataset, which comprises more than 1 billion masks derived from 11 million licensed and privacy-respecting images. The SA-1B dataset is notably fair, ensuring balanced and unbiased data collection. Trained on this extensive and diverse dataset, SAM exhibits remarkable robustness and generalization capabilities, sparking significant interest among researchers aiming to explore its potential across various real-world applications and to enhance its segmentation efficiency and accuracy.

Figure 3.3: Segment Anything Model (SAM) [32].



(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**)

3.2.0.1 Task

SAM aims to generate accurate segmentation masks for a wide variety of images by a flexible prompt-based segmentation approach. It allows users to provide prompts in the form of points, boxes, text or masks as illustrated in **Figure 3.3 (a)**, specifying what to segment in an image. The model is designed to be capable of handling ambiguous prompts and still producing a reasonable segmentation mask for at least one object in the image. This innovative approach involving prompt engineering enables SAM to generate accurate segmentation masks for new tasks or images without the need to retrain or fine-tune the model, resulting in zero-shot generalization, a powerful capability absent in traditional segmentation models.

3.2.0.2 Model

SAM consists of three main components designed for prompt segmentation.

- **Image Encoder:** Driven by the goals of scalability and effective pretraining approaches, a Vision Transformer (ViT) pre-trained with a Masked Autoencoder (MAE) [22] is employed, with minimal adaptations to handle high-resolution inputs. The encoder runs once per image and generates a powerful image embedding, which serves as an initial and fundamental step before prompting the model with downstream tasks.
- **Prompt Encoder:** It processes various types of sparse prompts (points, boxes, or text) and dense prompts (masks), converting them into embeddings. Sparse prompts are represented using positional encoding combined with learned embeddings, whereas dense prompts are embedded via convolutions. This encoder is lightweight and flexible, enabling SAM to work with diverse user inputs.
- **Mask Decoder:** It generates segmentation masks by combining image embeddings, prompt embeddings, and an output token. Using a modified Transformer decoder, it applies self-attention and cross-attention to refine embeddings in both prompt-to-image and image-to-prompt directions. After two decoding blocks, the image embedding is up-sampled and an MLP maps the output token to a dynamic linear classifier, which computes the foreground probability for each pixel, producing the final mask.

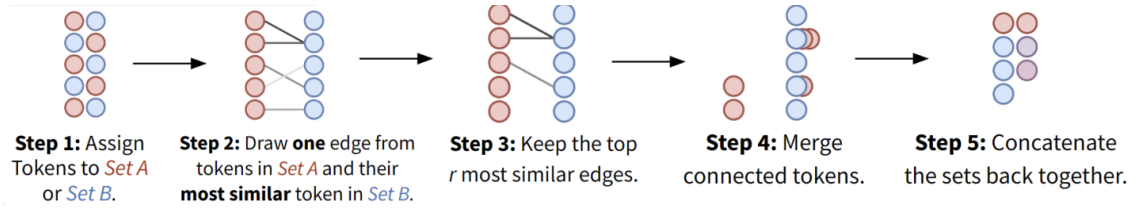
3.3 Token Merging Strategies

Token merging refers to the process of reducing the number of tokens processed in a Vision Transformer model by merging redundant or less informative tokens. The primary motivation behind token merging is to increase throughput and efficiency without the need for retraining the model. This is achieved by selectively merging tokens within each block of the transformer architecture, reducing the total number of tokens by a factor r. By adjusting r, we can strike a balance between speed and accuracy—fewer tokens lead to higher throughput but may decrease the model's accuracy. This section will explore the concept of BSM-based token merging in detail, discuss various algorithms for its application, and highlight their pros and cons in terms of computational efficiency and model performance.

3.3.1 Bipartite-Soft-Matching (BSM)

The goal of designing a reliable token merging algorithm is to effectively and efficiently identify similar tokens and aggregate them in a reasonable manner. This results in reduced token sequences that can be seamlessly and continuously passed through ViT blocks. The challenge lies in determining which tokens should be merged with which others in order to reduce the total number of tokens, while ensuring that the process itself remains computationally efficient. Although potential solutions, such as k-means clustering [49] or graph cuts [7], could be applied, these iterative methods are difficult to parallelize and are not suitable for batch processing, as clustering can merge an arbitrary number of tokens into a single group, making them impractical for this task. Since token matching may need to be performed multiple times within the network, potentially involving thousands of tokens, the runtime must be negligible, something that is not feasible with most iterative-based algorithms.

Figure 3.4: Bipartite-Soft-Matching (BSM) [3].



To address these challenges, a more efficient solution is proposed using **bipartite soft matching (BSM)**, whose process follows these steps (see visualization in **Figure 3.4**):

1. **Partitioning Tokens**: The tokens are split into two sets, $\mathbb{A}$ (src) and $\mathbb{B}$ (dst).
2. **Edge Creation**: Each token in set $\mathbb{A}$ is matched to its most similar (e.g., by cosine similarity) token in set $\mathbb{B}$, forming an edge between the two.
3. **Edge Selection**: The top r most similar edges are selected.
4. **Token Merging**: Tokens connected by the selected edges are merged, typically by averaging their embedding vectors.

5. **Reconstruction**: The remaining tokens in the two sets are concatenated back together.

This process results in a *bipartite graph* [1], where each token in set $\mathbb{A}$ has only one outgoing edge, making the identification of connected components during merging straightforward. This approach is computationally efficient, nearly as fast as randomly dropping tokens, and requires minimal implementation effort, offering a scalable and effective solution for token merging.

3.3.1.1 Computational Complexities

Processing Step	**Computational Complexity**
1. Partitioning Tokens	$O(N)$
2. Edge Creation	$O(N \cdot d' + A \cdot B \cdot d' + A \cdot B)$
3. Edge Selection	$O(A \log A)$
4. Token Merging	$O(r \cdot d)$
5. Reconstruction	$O(N \cdot d)$

- N: Number of tokens in the input sequence.
- d: Dimension of the original token embeddings corresponding to image patches, used in merging and subsequent reconstruction.
- d': Dimension of the feature space utilized for comparing token similarity, typically much smaller than d, i.e., $1 \ll d' \ll d$.
- A: Size of source set $\mathbb{A}$.
- B: Size of destination set $\mathbb{B}$.
- r: Number of tokens to be reduced (note: here r is defined as an absolute number, not as a ratio).

Table 3.2: Computational Complexities of Default BSM-based Token Merging Methods.

The computational complexity of BSM-based methods is primarily dominated by the second step, which involves identifying the most similar token pairs. This step results in a computational cost that scales quadratically with the number of tokens. Specifically, assuming there are N feature tokens whose dimension is d', the first step, token partitioning under the default setting, requires $O(N)$ operations to determine the location of each token. In the second step, for each token in the set $\mathbb{A}$, a comparison is made with every token in the set $\mathbb{B}$, leading to $O(N \cdot d' + A \cdot B \cdot d')$ computational cost, where the first term accounts for normalization and the second term accounts for pairwise similarity computation. Here, A and B denote the sizes of sets $\mathbb{A}$ and $\mathbb{B}$, respectively, with $A + B \leq N$. Subsequently, to find the maximum similarity for each token in $\mathbb{A}$ across the tokens in $\mathbb{B}$, a linear scan over $\mathbb{B}$ entries is required for each token in $\mathbb{A}$, resulting in an additional

$O(A \cdot B)$ computational cost. In the third step, selecting the top r most similar edges necessitates sorting, with a complexity of $O(A \log A)$. Once the most similar token pairs are well prepared as merge candidates, the subsequent steps—merging and reconstruction, follow straightforward procedures that do not introduce additionally significant computational overhead. The computational complexity of executing standard BSM-based token merging is summarized in **Table 3.2**.

3.3.2 Algorithms

Partitioning the tokens into two disjoint sets during the first step plays a crucial role in mitigating computational complexity. It reduces the need for exhaustive pairwise similarity computations among all tokens and subsequently decreases the computational burden in the max-selection and sorting steps. By dividing the tokens into smaller subsets, the total number of comparisons is significantly lowered, as the tokens in set $\mathbb{A}$ are compared only with the tokens in set $\mathbb{B}$. To further optimize the token merging process, various algorithms have been proposed for token partitioning. These methods aim to intelligently divide the tokens in a way that maximizes the likelihood that tokens in set $\mathbb{A}$ are able to find their most similar counterparts in set $\mathbb{B}$, thereby enhancing the efficiency and effectiveness of the matching process. In this section, we will discuss strategic token partitioning introduced by *ToMe* [3], *ToMeSD* [5] and *PiToMe* [75].

3.3.2.1 ToMe & ToMeSD

Both *ToMe* and *ToMeSD* adopt grid-based sampling strategies, wherein a spatially uniform partition inherently assumes that neighboring tokens convey similar information. This assumption ensures a sufficient number of representative tokens and raises the possibility that most tokens from the source ($\mathbb{A}$) can find highly similar matching counterparts in the destination ($\mathbb{B}$). This structured approach preserves spatial coherence while minimizing redundancy.

ToMe introduces an alternating partitioning strategy to ensure an equal division of tokens (see **Figure 3.5 (a)**). Specifically, tokens with even indices are assigned to set $\mathbb{A}$, while tokens with odd indices are assigned to set $\mathbb{B}$. This approach inherently imposes an upper limit on the merge rate, as no more than 50% of the tokens can be merged in a single step. If 50% of the tokens are merged, the entirety of set $\mathbb{A}$ is merged into set $\mathbb{B}$, leaving only every second column remaining and consequently halving the image resolution along the rows. However, because *ToMe* was designed for gradual merging with a controlled and moderate merge rate, this limitation does not present a significant issue.

ToMeSD adapts token merging for Stable Diffusion [63], where token merging is applied directly before the self-attention mechanism, and unmerging occurs immediately after self-attention (further explanations are provided in section **3.3.3 Placement of Token Merging Modules**). This approach necessitates a more aggressive merge rate to ensure computational efficiency without compromising generation quality. To achieve this, *ToMeSD* introduces an enhanced grid-based sampling approach with a randomness setting. In **Figure 3.5 (b)**, the model incorporates a 2D stride to form a structured grid, always selecting the upper-left corner as the destination token, while the remaining tokens are assigned to the source set. This method enforces a uniform grid partition but without introducing randomness, as set $\mathbb{B}$ remains identical for all inputs. On the other hand, randomly selecting destination tokens without enforcing spatial requirements (**Figure 3.5 (c)**) was also

explored. However, this approach can lead to undesirable clumping of destination tokens, which, according to experimental results, results in a significant performance drop for image synthesis. A reasonable solution combines the two strategies (**Figure 3.5 (d)**): randomly selecting one token as the destination token within each 2×2 region. This approach introduces sufficient randomness while maintaining a fair spatial distribution, which is crucial for diffusion-based models.

Figure 3.5: Token Partitioning Methods in *ToMe* and *ToMeSD* [5].

3.3.2.2 PiToMe

ToMeSD sought to enhance the alternating indexing strategy used in *ToMe* in order to increase the token reduction rate, whereas *PiToMe* identified an additional critical limitation in the original algorithm. The primary objective of *ToMe* is to efficiently detect and merge similar or redundant tokens. However, when alternating indexing is applied without supplementary contextual information, it risks merging highly informative tokens (e.g., those corresponding to edges or corners), potentially degrading model performance. To mitigate this issue, *PiToMe* introduced an **energy score**, inspired by graph energy in spectral graph theory [2], to rank token importance. This score is subsequently used to identify and preserve critical, information-bearing tokens during the partitioning process.

Large clusters of similar tokens, such as those representing background regions, are expected to exhibit high energy and are therefore considered suitable for merging. In contrast, smaller and more distinct regions, such as foreground elements, tend to have lower energy and should be preserved during the merging procedure. The proposed energy term operates on a fully connected graph constructed from the set of input tokens, evaluating the similarity between connected tokens based on predefined thresholds. This formulation ensures that the energy score is context-aware, capturing the semantic relationships among tokens.

Subsequently, based on the desired reduction rate, only tokens with the highest energy scores are selected for merging, thereby safeguarding essential information. During the merging process, energy scores are further utilized to partition tokens with similar energy levels into two sets, $\mathbb{A}$ and $\mathbb{B}$, using the alternating indexing strategy originally proposed in *ToMe*. This approach also imposes a theoretical upper bound on the merge rate, specifically at 50%. As a result, tokens in set $\mathbb{A}$ are highly likely to find semantically compatible counterparts in set $\mathbb{B}$. These matched tokens are then aggregated to form new token representations.

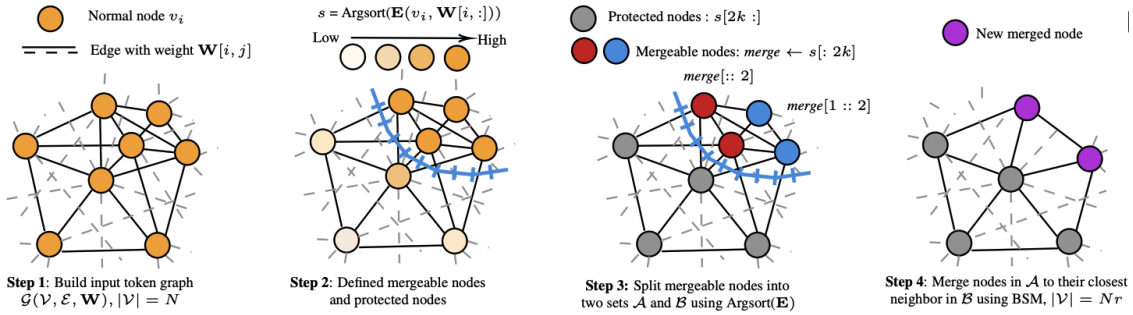


Figure 3.6: *PiToMe*: Energy-Score-Based Token Partitioning [5].

To formally present the complete process by which the energy score is employed as a heuristic for token partitioning in *PiToMe*, each stage involved in its computation and application is outlined below. A corresponding visualization illustrating this process is provided in **Figure 3.6**.

1. **Token Graph Construction.**
   Given a set of N token inputs denoted as $\hat{X}^l$ at layer l, a fully connected weighted graph $G = (V, E, W)$ is constructed, where:

   - V is the set of $N = |V|$ nodes, each corresponding to a token,
   - E is the set of $M = |E|$ edges, where each node is connected to all other nodes in the graph,
   - $W \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix representing pairwise edge weights.

   For the node features, the key vectors can be used optionally: $K = X^l W_K \in \mathbb{R}^{N\times d_k}$, where W_K is the learnable projection matrix, and each node $v_i \in V$ is associated with a feature vector of d_k dimensions.

   The edge weight $W[i,j]$ between node v_i and v_j is then computed by cosine distance: $W[i,j] = 1 - \cos(v_i, v_j)$, where $\cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\|\,\|v_j\|}$ for all $v_i, v_j \in V$.

2. **Energy Score Computation.**
   The energy score E_i for each node v_i is computed to quantify the redundancy of its local neighborhood. This approach is inspired by the concept of graph energy from spectral graph theory [2], where energy is defined as the sum of the absolute eigenvalues of the adjacency matrix W. The energy is defined as:

$$E_i = \frac{1}{N}\sum_{j\in\mathcal{N}(i)} f_m(\cos(v_i, v_j)), \qquad f_m(x) = \begin{cases} x, & x \geq m \\ \alpha(\exp(x-m)-1), & \text{otherwise} \end{cases}$$

   where m is a dynamic margin that is dependent on the layer, and α is a smoothing coefficient for the ELU [15] activation function. This function prioritizes highly similar neighbors while attenuating the influence of dissimilar ones. It also smooths the activation curve for negative inputs. Consequently, the energy score remains contextually aware.

3. **Sorting and Token Selection.**
   All tokens are sorted in descending order of their energy scores. Based on the target token reduction rate r, the top $(2 \cdot N \cdot r)$ tokens are selected as candidates for merging, and the rest are preserved to retain informative content:

   $$\texttt{s} = \texttt{argsort}(E, \texttt{descending=True}), \quad \texttt{merge} \leftarrow \texttt{s}[:2 \cdot N \cdot r], \quad \texttt{protect} \leftarrow \texttt{s}[2 \cdot N \cdot r :].$$

4. **Energy-Guided Partitioning.**
   The mergeable token set is further divided into two disjoint subsets $\mathbb{A}$ and $\mathbb{B}$ by alternating indices (i.e., odd and even positions) in the sorted list of merging candidates. This exploits the assumption that adjacent tokens in the sorted order are likely to have originated from the same semantic region, making them suitable for matching.

   $$\texttt{src} \leftarrow \texttt{merge}[::2], \quad \texttt{dst} \leftarrow \texttt{merge}[1::2].$$

5. **Bipartite Soft Matching and Token Merging.**
   For each token in $\mathbb{A}$, the best matching token from $\mathbb{B}$ is selected using cosine similarity. The matched pairs are then merged by aggregating their feature embeddings, such as by taking the average.

PiToMe is computationally efficient as it builds upon the BSM framework, ensuring fast processing through its bipartite matching approach. By incorporating an energy score, *PiToMe* enhances token partitioning by integrating the semantic meaning of tokens. This enables the model to prioritize the merging of similar tokens while protecting important, contextually significant tokens, thereby improving both computational efficiency and the preservation of essential features in ViT models.

3.3.3 Placement of Token Merging Modules

After designing the efficient and effective token merging algorithm, identifying the optimal insertion point within the Vision Transformer (ViT) architecture becomes the next critical consideration. There are two primary strategies for integrating the token merging module, as illustrated in **Figure 3.7**.

- **Insertion between the Attention and MLP blocks**

  The first approach, originally suggested by *ToMe*, inserts the token merging module between the Attention and MLP layers (**Figure 3.7 (a)**). This configuration is commonly applied to plain ViT models and offers practical advantages. By placing the merging module at this location, subsequent transformer layers process fewer tokens, resulting in computational savings and improved inference speed.

  One major benefit of this placement is its compatibility with the residual connection structure in the transformer blocks. Specifically, the Attention block resides within the first residual connection, which expects the original token sequence, while the MLP block is part of the second residual connection, which operates on the reduced token sequence. Inserting the merging module between these two stages ensures that each residual path receives appropriately formatted inputs, avoiding the need for additional architectural adjustments.

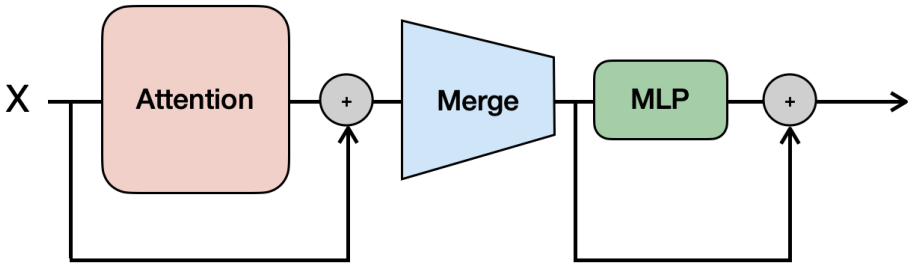
Another advantage of this placement is the availability of a meaningful metric for comparing token similarity. Since the merging occurs after the Attention block, the key matrices K, computed during attention, can naturally serve as the token feature representation. These keys encapsulate the semantic content of each token, making them well-suited for guiding the merging process.

- **Insertion directly before and after the Attention block**

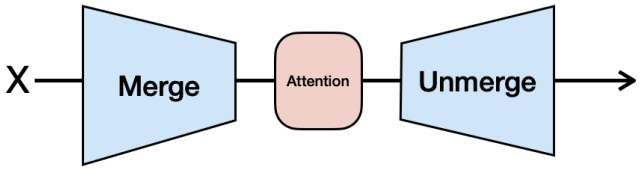
  The second approach positions the token merging module before the Attention mechanisms and includes an unmergeing operation immediately after it (**Figure 3.7(b)**). This strategy is often adopted for non-standard ViT architectures or models that incorporate attention mechanisms but deviate from the ViT structure. The aim in this scenario is to optimize the computational efficiency of the Attention block specifically, which often constitutes a significant computational bottleneck. To achieve meaningful acceleration, a relatively aggressive token reduction rate is typically required.

  For instance, Segformer++ [30] utilizes the Segformer [82] backbone, which inserts convolutional layers between vision transformer blocks. These convolutional layers necessitate spatial grid structures, thereby mandating the reconstruction of the original token sequence from the reduced representation. Similarly, *ToMeSD* is designed for use within the Stable Diffusion model [63], which employs a U-Net [64] architecture with symmetric downsampling and upsampling stages. These stages also require the spatial grid to function correctly. As a consequence, *ToMeSD* adopts the second placement choice and further refines the sampling strategy to support a higher reduction rate.

Figure 3.7: Architectural Considerations for Merge Module Placement within ViT.



(a) "`Merge`" between the Attention and MLP modules



(b) "`Merge`" before Attention, "`Unmerge`" immediately after Attention

3.3.4 Unmerge Operation: Model- and Task-Specific Considerations

The necessity of incorporating an unmerging module is inherently both task-dependent and model-dependent. The token merging strategy was initially introduced in the context of standard Vision Transformer (ViT) architectures applied to image classification tasks, where the final prediction is derived solely from the class token (`[CLS]`). In this setting, reducing the token sequence through merging does not interfere with the model's functionality, as the reduced sequence remains compatible with the model architecture. Therefore, this downsampling process does not hinder the model's ability to generate the final prediction, as the class token (`[CLS]`) remains sufficient for decision-making. However, in dense prediction tasks such as semantic segmentation, where the objective is to produce pixel-wise predictions, preserving spatial fidelity becomes crucial. In such cases, an unmerging operation is essential for reconstructing the original spatial arrangement of tokens from their reduced representations. This can be achieved simply by copying the feature embedding vectors of the reduced tokens to the tokens they were merged with. This restoration ensures that the model retains the capacity to make accurate, spatially resolved predictions across the full image domain.

3.4 Image Gradient in Traditional Image Processing

In traditional image processing, the *image gradient* is a fundamental concept used to measure the change in intensity or color [21, 69]. It is particularly useful for edge or corner detection, feature extraction, and texture analysis. The gradient at a given pixel indicates the direction and rate of the most significant intensity change in its local neighborhood.

3.4.1 Definition and Interpretation

Mathematically, the image gradient is a vector defined as the partial derivatives of the image intensity function $I(x, y)$ with respect to spatial coordinates:

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

The magnitude of the gradient vector $\|\nabla I\|$ describes how steep the intensity change is, while the orientation $\theta = \arctan\left(\frac{\partial I/\partial y}{\partial I/\partial x}\right)$ indicates the direction of the change. High gradient magnitudes typically correspond to edges or boundaries between objects.

3.4.2 Numerical Approximation

In discrete images, derivatives are typically approximated by *finite differences* [37]. Two common first-order numerical methods are the *central differences* and the *Sobel operator*.

3.4.2.1 Central Differences

The central difference method uses symmetric neighboring pixels to approximate derivatives. For a pixel at (x, y), the partial derivatives are approximated as:

$$\frac{\partial I}{\partial x} \approx \frac{I(x+1, y) - I(x-1, y)}{2}, \quad \frac{\partial I}{\partial y} \approx \frac{I(x, y+1) - I(x, y-1)}{2}.$$

These can be represented using convolution with the following kernels:

$$K_x^{\text{central}} = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad K_y^{\text{central}} = \frac{1}{2}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}.$$

3.4.2.2 Sobel Operator

The Sobel operator extends the central difference method by incorporating a smoothing effect perpendicular to the derivative direction. The Sobel kernels are defined as:

$$K_x^{\text{sobel}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad K_y^{\text{sobel}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

These kernels are derived by applying a 1D smoothing filter orthogonal to the differentiation direction. For example, K_x^{sobel} is obtained by combining a horizontal gradient kernel $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ with a vertical smoothing filter $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$.

Central Differences vs. Sobel Operator

The central difference method is computationally efficient and easy to implement, but it is highly sensitive to noise. In contrast, the Sobel operator offers greater robustness due to its smoothing component (see comparison in **Figure 3.8**), which helps suppress high-frequency noise and provides more stable gradient estimation in real-world images.

Thus, the Sobel operator is often preferred in practical applications such as edge detection and object contour extraction. It strikes a balance between accurate gradient estimation and noise suppression, making it more effective in complex visual environments.

Figure 3.8: Central Differences vs. Sobel Operator.

4 Methods

This section presents the step-by-step development of our approach about integrating token merging algorithms into the Segment Anything Model (SAM). **Section 4.1** provides an in-depth analysis of SAM's sophisticated model architecture and examines the potential challenges and risks associated with incorporating token merging. **Section 4.2** outlines the experimental setup. With a comprehensive understanding of SAM, **Section 4.3.1** focuses on adapting existing token merging algorithms, detailing specific design considerations and configurations. **Section 4.3.2** presents the main experimental results along with a discussion and analysis of the observed effects of applying token merging within SAM. In the final stage, we introduce *GradToMe* in **Section 4.4**, our proposed method developed in response to the limitations identified in prior approaches, supported by relevant experimental evidence.

4.1 In-Depth Analysis of SAM's Architecture

To effectively integrate token merging algorithms into SAM, it is crucial to first gain a deeper understanding of its architecture. By analyzing the structure and functionality of SAM, we can identify the most suitable approach for incorporating token merging while maintaining segmentation quality. Furthermore, drawing insights from previous studies helps us determine whether existing methods can be directly applied or if adaptations are needed to align with SAM's unique design. If modifications are required, we must explore strategies to refine these methods, ensuring seamless integration and upholding SAM's core design concept to the greatest extent.

4.1.1 Model Size Analysis

We conducted a detailed parameter analysis of SAM across different backbone variants (ViT-b, ViT-l, and ViT-h) to evaluate the distribution of computational resources among the three main components: **image encoder**, **prompt encoder**, and **mask decoder**. The results are summarized in **Table 4.1**.

Table 4.1: Parameters of SAM Variants (in Millions) by `torch.numel()`.

Backbone	Total	Image Encoder	Prompt Encoder	Mask Decoder	Image Encoder %
ViT-b	93.74	89.67	0.0062	4.06	95.7%
ViT-l	312.34	308.28			98.7%
ViT-h	641.09	637.03			99.4%

The image encoder constitutes the vast majority of the total parameters in all SAM variants, consistently exceeding 90% of the model size and serving as the primary computational bottleneck. In contrast, the prompt encoder remains extremely lightweight, with only 6,220 parameters across all models, contributing negligibly to the overall computational cost. The mask decoder, while larger than the prompt encoder at a fixed size of 4,06M parameters, is still significantly smaller than the image encoder. Furthermore, as the model scales from ViT-b to ViT-h, the increase in total parameters is entirely attributed to the image encoder, further emphasizing its dominance in computational complexity.

In an encoder-decoder architecture, the encoder is designed to be computationally intensive because its primary role is to extract rich, high-level feature representations from the input data, capturing complex patterns, structures, and relationships essential for downstream tasks. This requires a significant amount of parameters and multiplications to handle complicated visual information, including texture, object boundaries, and contextual relationships. In contrast, the decoder is typically much lighter, as its main function is to process and refine the encoded features into the final output, such as the segmentation mask or the class label. Since it operates on pre-extracted features, its task is relatively simpler and less resource-intensive. This architectural choice enhances model performance while maintaining computational efficiency by concentrating the majority of computation in the encoder, allowing the decoder to operate more lightly and task-specifically. Moreover, SAM's robust and highly capable image encoder demonstrates impressive zero-shot transferability, as evidenced by its performance across a range of tasks such as edge detection, instance segmentation, and object proposal generation [32].

SAM's inefficiency essentially arises from its heavy image encoder, making it hard for real-time deployment on mobile devices with limited computing resources and thus the natural target for optimization. A straightforward yet effective approach to accelerate SAM is to replace its encoder with more efficient backbones, which has been explored and validated in different ways by numerous researchers. MobileSAM [95] achieved this by involving distilling knowledge from the original large encoder to the smaller one while maintaining compatibility with the existing mask decoder. EfficientSAM [83] introduces SAM-leveraged Masked Image Pretraining (SAMI), which utilizes the ViT-H encoder of SAM to generate feature embeddings and train lightweight ViT encoders to reconstruct these features instead of image patches. The pretrained lightweight encoders are then fine-tuned with SAM's decoders for the segment anything task, maintaining strong performance with improved efficiency. The lite feature extractor of RAP-SAM [84] improves efficiency by adopting lightweight backbones like ResNet18 [23] and TopFormer [99], and incorporates a Feature Pyramid Network(FPN) [45] with deformable convolution to enhance feature alignment and multi-scale fusion in real time.

In our task, token merging algorithms are integrated into SAM without training to reduce the number of redundant tokens processed by the encoder, thereby decreasing computational load without compromising the quality of extracted features. Since the encoder is the most computationally demanding component, optimizing it is crucial, and it is equally important to ensure that the mask decoder's performance remains intact. Although the decoder is also ViT-based and suffers from quadratic computational complexities due to self-attention and cross-attention mechanisms, we chose not to apply token merging to the mask decoder, as its size and computational resource requirements are minimal in comparison to the overall model, and we did not want to compromise

its decision-making capability. Moreover, the mask decoder's cross-attention mechanism plays a critical role in fusing information from both the image encoder and the prompt encoder, ensuring accurate and context-aware segmentation.

4.1.2 SAM's Image Encoder: Advanced ViT

In the previous section, we identified the image encoder as the primary computational bottleneck in SAM, making it the optimal target for efficiency improvements through token merging strategies. In this section, we provide an in-depth analysis of SAM's image encoder, with a particular focus on its key modifications compared to a standard Vision Transformer, as shown in **Figure 3.1**. SAM's image encoder is not a conventional ViT, it incorporates three critical modifications: **periodic windowed-global attention**, **decomposed relative positional embeddings**, and the **convolutional neck**. These design choices significantly contribute to SAM's capability as a powerful foundation model. Nevertheless, these modifications also introduce challenges in directly applying existing token merging techniques that have been effectively validated in other plain ViT-based models, necessitating alternative approaches to adapt previous proven ideas and methods. This section explores these architectural modifications in detail and discusses key considerations for incorporating token merging while preserving the model's effectiveness.

4.1.2.1 Periodic Windowed-Global Attention

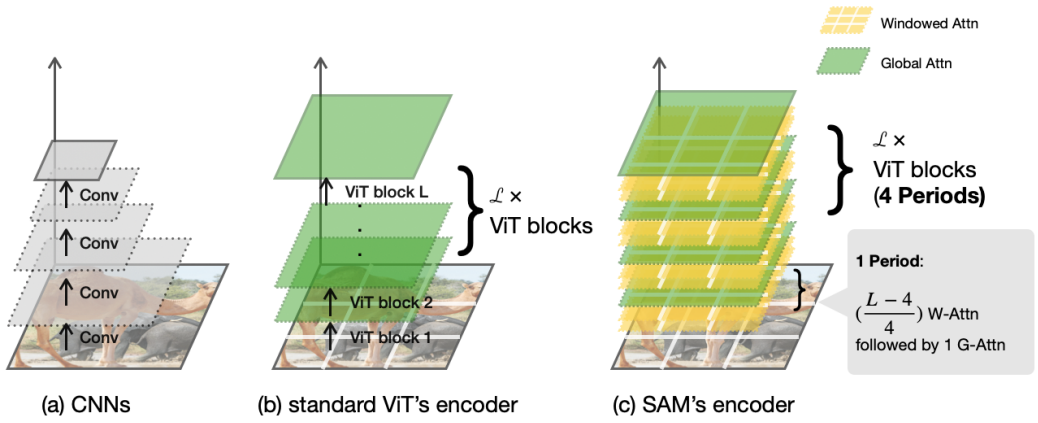


Figure 4.1: Comparison of Different Architectures for Obtaining Feature Embeddings.

(a) **CNN**: Extracts feature embeddings through a feature pyramid, capturing hierarchical spatial information. (b) **standard ViT's encoder**: Applies global attention in all layers, enabling long-range dependencies but at high computational cost. (c) **SAM's encoder**: Uses periodic windowed-global attention, balancing efficiency and global context.

Feature extraction has long been a fundamental aspect of computer vision models, with Convolutional Neural Networks(CNNs) historically serving as the dominant architecture. CNNs efficiently capture spatial hierarchies through local receptive fields and weight sharing, introducing an inherent

inductive bias that enhances data efficiency. This inductive bias includes *locality*, which focuses on neighboring pixels, *translation invariance*, which ensures feature recognition regardless of position, and *hierarchical feature learning*, where deeper layers progressively capture more abstract patterns. These properties make CNNs highly effective, especially with limited training data. However, CNNs struggle to model long-range dependencies, as their receptive field expands only gradually through deeper layers.

To address this limitation, Vision Transformers (ViTs) have emerged as a powerful alternative. Unlike CNNs, ViTs process images as sequences of patches and utilize self-attention mechanisms to establish long-range dependencies from the outset. This ability to capture global context is a significant advantage over CNNs. However, standard ViTs lack the inductive bias inherent to CNNs, making them less data-efficient and requiring extensive pretraining on large datasets to generalize well. Additionally, their global self-attention mechanism incurs a quadratic computational complexity of $O(N^2)$, where N represents the number of tokens. This computational burden becomes prohibitive for high-resolution images, limiting the scalability and real-time deployment of ViTs.

To enhance the efficiency of ViT-based models while maintaining their ability to capture long-range dependencies, SAM introduces a novel **periodic Windowed-Global attention** mechanism (**Figure 4.1 (c)**) in its image encoder. Instead of applying global attention across all layers, SAM strategically interleaves windowed attention and global attention blocks. Windowed attention, already adopted in several ViT variants, has demonstrated strong performance in prior research [12, 34, 48, 70, 85, 98]. A notable example of its success is the Swin Transformer [48], which further refines this approach by introducing a hierarchical structure with shifted windows, enabling both efficient local attention and effective global context modeling, thus improving performance across various vision tasks. By restricting self-attention computations within localized windows as shown in **Figure 4.2**, the quadratic complexity $O(N^2)$ is reduced to $O(num\_windows \cdot w^2)$, where w is the window size.

Although windowed attention is computationally efficient, it introduces a significant limitation: it restricts information flow across windows unless additional strategies, such as the hierarchical shifted windows used in Swin Transformer, are applied. Since each token only interacts with others within its local window, the model loses the global receptive field that standard ViTs provide. To overcome this, SAM periodically incorporates global attention blocks, allowing tokens to interact across windows. This strategic alternation enables SAM's encoder to maintain the efficiency benefits of windowed attention while periodically restoring long-range dependencies through global attention. By facilitating cross-window interactions, global attention compensates for the local constraints of windowed attention, ensuring that the model captures both fine-grained details and holistic contextual relationships.

Furthermore, this hybrid attention mechanism helps mitigate the lack of inductive bias in standard ViTs. By introducing structured locality through windowed attention and periodically reinforcing global context, SAM's design implicitly incorporates hierarchical feature extraction akin to CNNs. This thoughtful balance between local and global attention enables SAM's image encoder to achieve strong performance without incurring the prohibitive computational costs associated with full global self-attention.

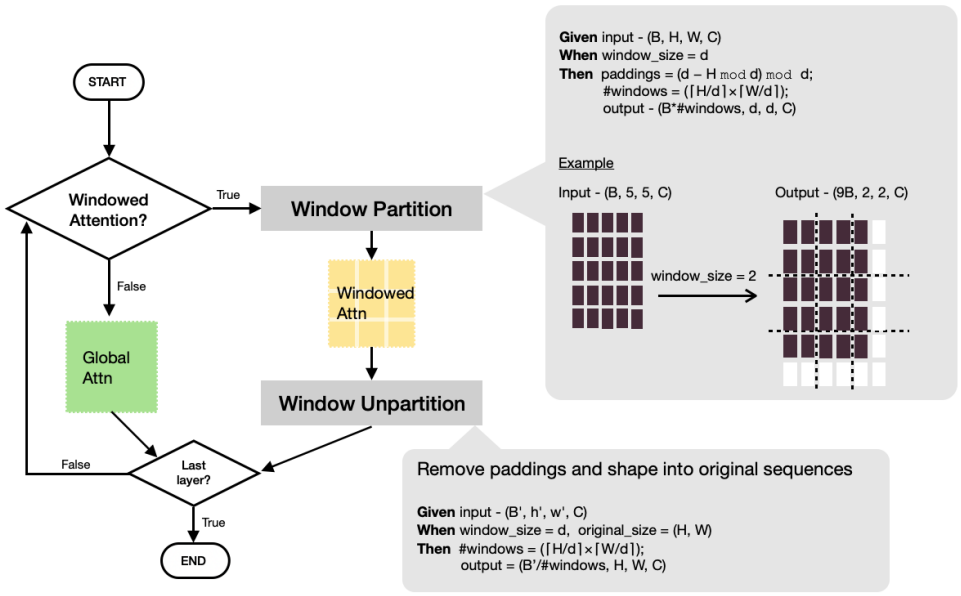


Figure 4.2: Windowed Attention in SAM.

Integrating Token Merging

- **Preserving Spatial Structure During Window Partitioning**
  - When applying **windowed** attention, token merging should ensure that spatial relationships within local windows are maintained or can be easily reconstructed.
  - Merging should avoid distorting the window alignment, as misaligned tokens could disrupt the effectiveness of local self-attention.
- **Ensuring Consistent Token Counts Across Windows for Efficient Batch Processing**
  - Since ViTs rely on fixed-sized token batches for efficient computation, the number of merged tokens within each window must remain consistent.
  - Without extra padding, token merging must be designed to ensure equal token counts across all windows, preventing irregular batch sizes that could hinder parallel processing.
- **Balancing Global Attention Efficiency and Spatial Relationship Preservation**
  - Although merging tokens in ViT layers with **global** attention significantly improves efficiency, only a few blocks perform **global** attention, allowing tokens to exchange information across windows. Therefore, careful consideration is required to balance efficiency and information flow.
  - A careful balance is required, reducing tokens enough to enhance throughput while retaining enough information to maintain **global** spatial coherence, excessive merging could weaken the model's ability to capture long-range dependencies.

4.1.2.2 Decomposed Relative Positional Embeddings (DRPE)

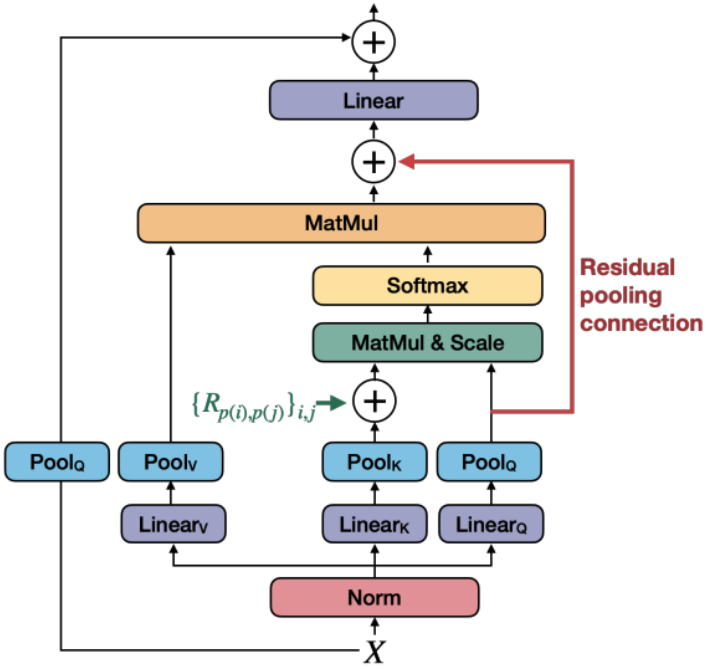


Figure 4.3: The improved Pooling Attention Mechanism proposed by MViT [40] incorporates Decomposed Relative Position Embeddings.

SAM employs **decomposed relative positional embeddings** in its self-attention mechanism to enhance spatial structure modeling. The idea of relative positional embeddings (RPE) was first introduced in NLP by Shaw et al. [67] to encode the relative distances between words. Later, MViTv2 [40] extended this concept to vision models and further proposed decomposing RPE along spatial dimensions to improve efficiency. The intuition behind decomposed RPE is to enhance *shift-invariance* in self-attention. Standard absolute positional embeddings encode fixed spatial locations, meaning that two patches with the same content may interact differently based on their absolute position. This contradicts the principles of visual perception, where relative positioning often carries greater significance than absolute placement.

To address this, SAM employs a decomposed formulation of RPE. Instead of computing a full pairwise embedding matrix for all tokens, SAM factorizes RPE along height and width axes:

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top + E^{(\text{rel})}}{\sqrt{d}}\right)V$$

where

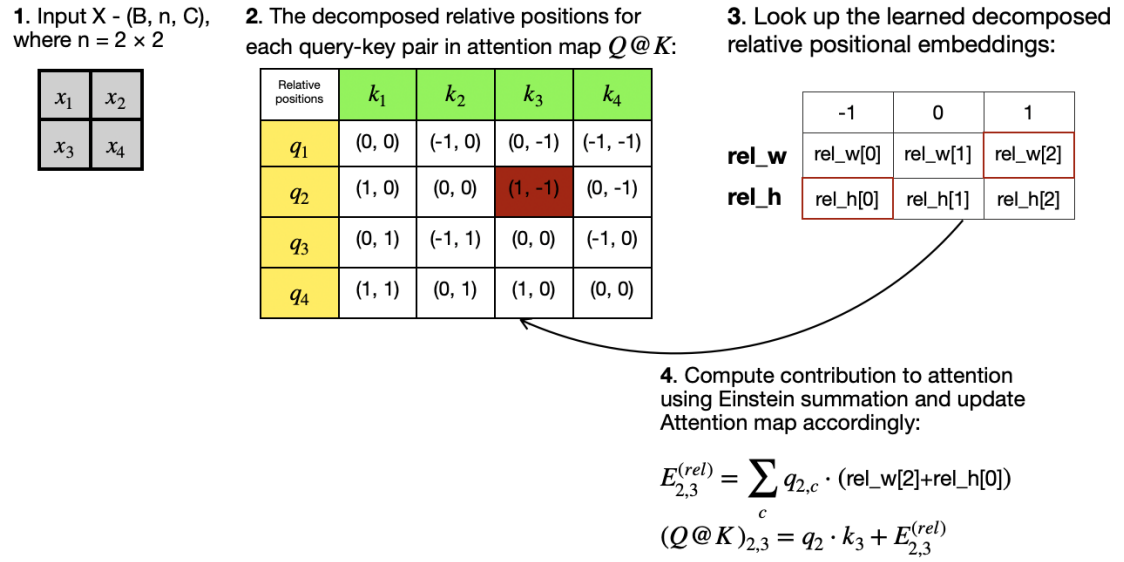
$$E^{(\text{rel})}_{ij} = Q_i \cdot R_{p(i),p(j)}$$

$$R_{p(i),p(j)} = R^{\text{h}}_{h(i),h(j)} + R^{\text{w}}_{w(i),w(j)}$$

where R^h and R^w represent relative positional embeddings for height and width, respectively. In **Figure 4.4**, we present a detailed illustration of this process through a concrete example. This significantly reduces computational complexity from $O(HW)$ to $O(H+W)$, where H and W denote the number of tokens along the height and width dimensions respectively, thereby enhancing scalability for high-resolution images.

This allows the attention mechanism to account for relative spatial relationships while maintaining computational efficiency. By employing decomposed RPE, SAM enhances both efficiency and robustness, preserving *shift-invariance* while maintaining rich spatial relationships in self-attention computation.

Figure 4.4: Example: Enhancing Attention Map with Decomposed Relative Positional Embeddings in SAM.



Integrating Token Merging Integrating token merging strategies into SAM presents unique challenges due to the deliberate design of its attention mechanism. Previous works [3, 5, 30, 54, 75] on token merging typically targeted models utilizing standard self-attention mechanisms without relative positional encoding. In these models, tokens just exchange information across all pairs, and their positional order is often disregarded when computing attention scores. This simplifies the merging process, as once similar patches are identified and merged, the reduced token sequence can be directly output and passed to the next step while remaining compatible with the model. However, SAM's attention mechanism, which employs RPE, requires additional considerations. Each patch in the reduced token sequence must retain its original position within the grid to preserve relative spatial relationships. Moreover, some patches in the merged sequence are aggregates of multiple original tokens, covering a larger spatial area. This complicates the definition of relative positions for query-key pairs, as the relative positional embeddings of merged tokens must be carefully recalibrated to reflect their new spatial context. Thus, while token merging improves computational efficiency, its integration with SAM's attention mechanism must be carefully designed to ensure that the model retains its ability to capture meaningful spatial relationships.

4.1.2.3 Convolutional Neck

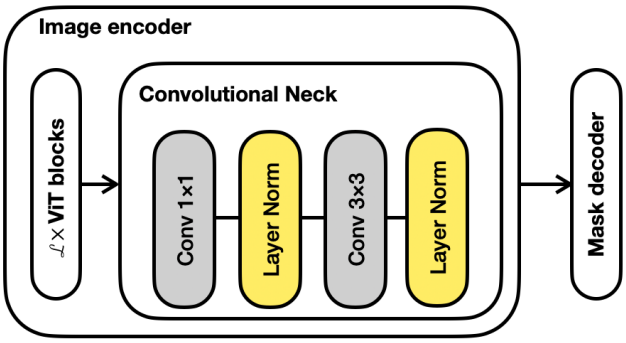


Figure 4.5: Convolutional Neck in SAM's Image Encoder.

The last notable deviation of SAM's image encoder from standard Vision Transformer encoders is the convolutional neck following all ViT blocks and preceding the decoder. Many ViT-based models extending CNN-based adapters [13, 82] to generate multi-scale representations that enhance performance, particularly for tasks such as segmentation, where multi-scale features and local processing are beneficial. SAM also introduces a convolutional processing stage(see **Figure 4.5**) to refine the extracted features before they are passed to the decoder. The convolutional neck consists of two convolutional layers interleaved with layer normalization: a 1×1 convolution followed by a 3×3 convolution, each accompanied by 2D layer normalization.

SAM's convolutional neck serves as a critical transition between the ViT backbone and the mask decoder, refining features extracted from the transformer blocks before passing them to the segmentation head. The first layer, a 1×1 convolution, performs channel-wise feature transformation by re-weighting and recombining information across channels without altering the spatial resolution. This is followed by a 3×3 convolutional layer, which enhances local feature representation by capturing spatial relationships among neighboring patches. Together, these layers refine the feature map, ensuring that the segmentation model preserves spatial integrity while benefiting from the rich, global contextual understanding provided by the ViT blocks. This architectural choice improves the quality of the predicted masks by improving the localization precision and facilitating smoother integration with the mask decoder.

Integrating Token Merging When integrating token merging into SAM, it is essential to consider its impact on the convolutional neck. Since convolutional operations rely on a structured grid representation, the reduced token sequence resulting from token merging must be appropriately recalibrated before entering the convolutional layers. This involves reconstructing the full spatial grid by unmerging the aggregated tokens, ensuring that the convolutional kernels can operate effectively without disrupting the spatial continuity required for feature refinement.

4.2 Experimental Setup

We conducted several experiments to evaluate the performance of the Segment Anything Model (SAM) enhanced with token merging modules, using **ViT-B** as the backbone and **without** additional training. Our main experimental setup was based on the configuration provided in HQ-SAM [29]. This section outlines the experimental setup, including the task definition, the dataset employed, the evaluation metrics, and a concise overview of the implementation details.

Task The task addressed in this study is *promptable segmentation*, in which SAM generates segmentation masks in response to user-provided prompts. To facilitate experimentation in this setting, each input image must be paired with a valid prompt. In our setup, bounding boxes are derived from the ground truth masks and passed into the prompt encoder. Specifically, for each binary mask, the bounding box is computed by identifying the extreme pixel coordinates (i.e., the minimum and maximum values along the x and y axes) that define the spatial extent of the foreground object. Furthermore, since SAM requires input images to have a fixed resolution of 1024×1024, all images are resized accordingly in data loaders to ensure compatibility. For the purpose of fair evaluation, the predicted segmentation masks are interpolated back to the original image dimensions prior to the computation of the evaluation metrics.

Dataset For a comprehensive assessment of the segmentation performance of our proposed model - SAM with token merging modules, we conduct experiments across a variety of datasets, including four highly detailed segmentation datasets: DIS [58] (validation set), ThinObject-5K [44] (test set), COIFT [44], and HR-SOD [93].

Evaluation Metrics To assess the effectiveness and efficiency of the proposed model, we employed a combination of widely recognized evaluation metrics: `Mask IoU`, `Boundary IoU`, and `FLOPs/im`. These metrics collectively enable a comprehensive analysis of segmentation accuracy and computational performance. `Mask IoU` is a standard metric for evaluating the quality of predicted segmentation masks. It quantifies the overlap between the predicted and ground truth masks by computing the ratio of the area of intersection to the area of union. `Boundary IoU`, places greater emphasis on the alignment of object boundaries, making it particularly suitable for evaluating fine-grained segmentation tasks where boundary precision is critical. To evaluate computational efficiency, we report the number of floating-point operations (FLOPs) required per input image (`FLOPs/im`). This metric reflects the total number of computational steps performed during a model's forward pass, providing insight into the resource demands of the model. An increased number of FLOPs generally reflects a model with greater computational complexity, which may enhance accuracy but also incurs higher demands in terms of computational resources and inference time. FLOP count analysis was conducted empirically by `fvcore.nn.FlopCountAnalysis` [57], which counts a single fused multiply-add operation as one FLOP. By comparing FLOPs before and after integrating token merging modules, we assess the extent to which our method reduces computational overhead.

Implementation All models and algorithms are implemented in `PyTorch` and executed on Apple's Metal Performance Shaders (MPS) backend. The Segment Anything Model (SAM) [32] serves as the baseline for evaluation. We referenced publicly available libraries for existing token merging approaches, including ToMe [3], ToMeSD [5], and PiToMe [75]. Our primary contributions include the integration of token merging modules into SAM and the enhancement of existing token merging algorithms through the development of *GradToMe*. The full implementation is publicly available at: https://github.com/xxjsw/tome_sam.

4.3 Integration of Token Merging Modules into SAM

This section outlines our efforts in incorporating existing token merging modules into the Segment Anything Model (SAM) to improve overall efficiency without compromising segmentation accuracy. Specifically, we tailor established token merging strategies, which are originally developed for conventional Vision Transformer (ViT) frameworks, to align with the architectural characteristics and operational demands of SAM, with a focus on promptable segmentation tasks.

4.3.1 Design Considerations and Integration Strategies

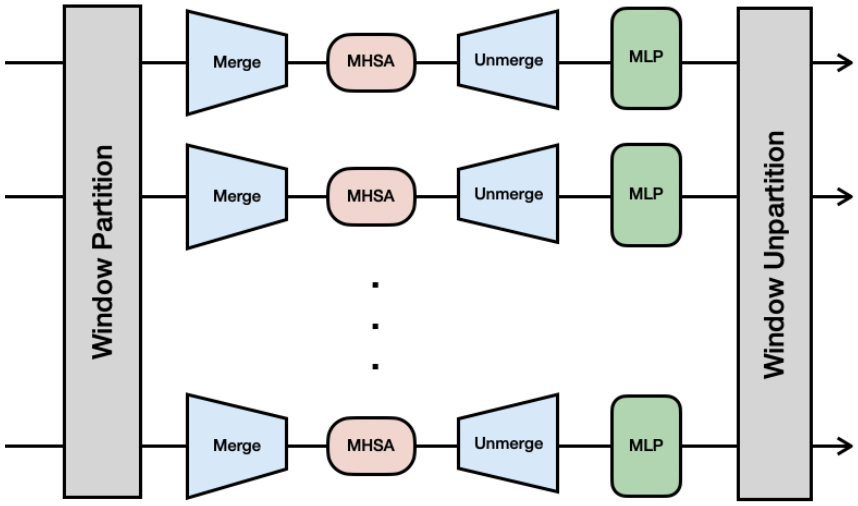
Token Merging Module Placement

As discussed in **Section 4.1.2**, SAM's image encoder differs from standard Vision Transformers due to critical modifications that may complicate the integration of token merging modules, as suggested by previous studies when the backbone is a plain ViT (**Figure 3.7 (a)**). Consequently, we chose to position the token merging module immediately before the Multi-Head Self-Attention (MHSA) block, with the unmerging module placed directly after it (**Figure 3.7 (b)**), such that the 2D structure is preserved throughout, except within the Attention block. This approach aligns with the recommendations from studies involving non-standard ViT backbones [30] or non-ViT architectures that incorporate attention mechanisms [5].

The "merge" and "unmerge" functions are simply defined as follows: Given two tokens, x_1 and x_2, with c channels in $\mathbb{R}$, and assuming that $x_1 \approx x_2$ (i.e., the tokens to be merged are similar or redundant), the merge operation combines them into a single token: $x^*_{1,2} = \texttt{merge}(x_1, x_2) = \frac{x_1+x_2}{2}$. Subsequently, the "unmerge" operation is defined as assigning the merged token $x^*_{1,2}$ to both tokens: $x'_1 = x'_2 = \texttt{unmerge}(x^*_{1,2}) = x^*_{1,2}$.

Although this operation results in some information loss, the tokens involved are already similar, so the error introduced is minimal. Furthermore, in layers employing **Windowed Attention**, token merging and unmerging should be applied for each window individually, as illustrated in **Figure 4.6**. This reduces the number of tokens within each window, which is beneficial in several ways. Specifically, token merging within each window can be batched and processed in parallel, increasing computational efficiency. Additionally, the reduced number of tokens per window decreases the complexity of token similarity comparisons and improves the likelihood of identifying the most similar tokens. This is due to the fact that token merging is now performed within smaller, spatially closer subsets.

Figure 4.6: Per-Window Merge and Unmerge Operations in a Single ViT Layer.



*Residual connections are omitted for clarity.

This positioning strategy is particularly well-suited for SAM's image encoder, as it ensures that the two critical modifications—**4.1.2.1 Periodic Windowed-Global Attention** and **4.1.2.3 Convolutional Neck**—remain unaffected. Tokens passed through these modules retain their original, full-grid token sequences. Consequently, the placement of the token merging module allows for targeted runtime optimization of the Attention block. However, this method does not progressively increase the number of merged tokens as originally intended in token merging for standard Vision Transformer models, it necessitates a relatively aggressive token reduction rate r.

Prompt-Specific Token Merging?

Given that SAM is designed for prompt-specific segmentation tasks, the integration of token merging modules raises the question of whether token merging should also be prompt-specific. After careful consideration, we choose not to adopt this approach. SAM's original architectural intent is to run the image encoder only once per image to generate general-purpose image embeddings. These embeddings can then be flexibly fused with various types of prompts in the mask decoder to produce corresponding segmentation outputs. This design enables the model to adapt to different prompt types and segmentation tasks without re-encoding the image, as image encoding is the most computationally intensive component.

To remain consistent with this architecture, and since our modifications apply solely to the image encoder, we implemented token merging in a prompt-agnostic manner. That is, token merging is applied during the encoding phase without consideration of any specific prompt. This preserves the core design philosophy of generating reusable image embeddings from a single forward pass, while improving efficiency by reducing tokens in selected layers. Although prompt-specific token

merging might further optimize computation for a given prompt, it would undermine the generality of the image embeddings and necessitate re-encoding the image for each prompt, which contradicts the foundational principle of SAM and introduces substantial computational overhead.

Merge Rate: Should It Be Adaptive?

To improve efficiency, we apply repeated token merging and unmerging around the Attention blocks in selected ViT layers, which necessitates a relatively aggressive token reduction rate r, defined as the percentage of tokens to be merged. Under default configurations, the maximum reduction rate for *ToMe* and *PiToMe* is 50%, as both methods utilize alternate indexing, where $r = 50\%$ implies that all tokens in set $\mathbb{A}$ are merged. In comparison, *ToMeSD* allows for a maximum reduction rate of 75% when using a 2×2 stride partition. This further raises the question of whether adaptive strategies should be employed to dynamically determine the token reduction rate, given that different input images may exhibit diverse structural characteristics. For instance, ALGM [54] introduces an adaptive merging scheme that selectively merges tokens within 2×2 windows when their similarity exceeds a predefined threshold in the initial ViT layer. While this method is straightforward and effective for plain ViT architectures, it poses challenges for integration into SAM due to its use of windowed Attention. Specifically, adaptive merging could result in varying token counts across different windows, thereby necessitating padding to preserve tensor dimensions during batch processing. In extreme cases, certain windows may retain nearly all tokens due to low inter-token similarity, leading to little or no computational savings after padding. Moreover, SAM is designed to support efficient batch inference—a critical requirement in real-world applications, such variability would likely undermine the benefits of token reduction. Consequently, to maintain compatibility with SAM's architecture and ensure stable and fair benchmarking, we choose to use a fixed token reduction rate and set it uniformly to the theoretical maximum of $r = 50\%$ across all evaluated methods.

We perform a computational complexity analysis to compare the Multi-Head Self-Attention (MHSA) block before and after integrating the token merging module, with a token reduction rate of $r = 50\%$. In our implementation, tokens are merged prior to the MHSA computation and unmerged immediately after it, specifically after the attention computation but before the final projection. Therefore, the overall computational complexity, including linear projections and the weighted summation of attention scores following the softmax operation, is given by $O(3Nd^2 + 2N^2d + hN^2)$, where N is the number of input tokens, d is the embedding dimension, and h is the number of attention heads (see **Section 3.1.3** for a detailed breakdown of self-attention complexity). When applying a token reduction rate of $r = 50\%$, we effectively replace N with $0.5N$ in the above expression, yielding a reduced complexity of $O(\frac{3}{2}Nd^2 + \frac{1}{2}N^2d + \frac{1}{4}hN^2)$.

Undoubtedly, token merging introduces additional overhead, particularly from the computation of pairwise cosine similarity scores. In **Section 3.3.1.1**, we provide a detailed breakdown of the computational complexity analysis for each step. For BSM-based token merging algorithms, let A and B represent the sizes of the token sets $\mathbb{A}$ and $\mathbb{B}$, respectively, and $A + B \leq N$. Let $d' \leq d$ denote the dimensionality of the feature used for similarity comparison, which could be reduced a lot via multi-head aggregation (see **Section 4.3.1 Handling Multi-Head Attention** for further discussion). The complexity of computing pairwise cosine similarity is therefore $O(Nd' + ABd')$, where the first term corresponds to the normalization step, and the second term arises from the dot product computation. In the case of *PiToMe*, which utilizes energy scores over a fully-connected

graph, pairwise similarities are computed for all token pairs, and the full similarity matrix is reused during matching, the complexity becomes $O(Nd' + N^2d')$. **Table 4.2** summarizes the complexity of different token merging methods under default setting.

Table 4.2: Computational Complexity of Different Token Merging Algorithms Under Default Settings.

Algorithm	A	B	Complexity
ToMe	$0.5N$	$0.5N$	$O(Nd' + \frac{1}{4}N^2d')$
ToMeSD	$0.75N$	$0.25N$	$O(Nd' + \frac{3}{16}N^2d')$
PiToMe	—	—	$O(Nd' + N^2d')$

To conclude the complexity analysis, we demonstrate that integrating token merging, even when employing the most computationally demanding variant, *PiToMe*, yields a net reduction in the computational complexity of each attention block. For a token reduction rate of $r = 50\%$, the MHSA complexity decreases from

$$C_{\text{orig}} = \underbrace{3Nd^2}_{\text{QKV projections}} + \underbrace{2N^2d}_{\text{Attention scores + Weighted sum}} + \underbrace{hN^2}_{\text{Softmax}}.$$

to

$$C_{\text{reduced}} = \underbrace{\frac{3}{2}Nd^2}_{\text{QKV projections}} + \underbrace{\frac{1}{2}N^2d}_{\text{Attention scores + Weighted sum}} + \underbrace{\frac{1}{4}hN^2}_{\text{Softmax}}.$$

Incorporating *PiToMe*'s overhead of $O(Nd' + N^2d')$, where $d' = \frac{d}{h}$ due to head aggregation, the total complexity becomes:

$$C_{\text{reduced\_total}} = \underbrace{\frac{3}{2}Nd^2}_{\text{QKV projections}} + \underbrace{\frac{1}{2}N^2d}_{\text{Attention scores + Weighted sum}} + \underbrace{\frac{1}{4}hN^2}_{\text{Softmax}} + \underbrace{N \cdot \frac{d}{h} + N^2 \cdot \frac{d}{h}}_{\text{PiToMe cost}}.$$

The improvement is characterized by the *reduction factor* $R = \frac{C_{\text{reduced\_total}}}{C_{\text{orig}}}$, which exhibits distinct scaling regimes:

- **Large-N (Global Attention):**
  The N^2dominant terms yield:

$$R_{\text{global}} \approx \frac{\frac{1}{2}N^2d + \frac{1}{4}hN^2 + N^2 \cdot \frac{d}{h}}{2N^2d + hN^2} = \frac{1 + \frac{h}{2d} + \frac{2}{h}}{4 + \frac{2h}{d}} \xrightarrow{d \gg h, h \gg 1} \frac{1}{4}$$

possibly yielding up to a 4× speedup when the embedding dimension d dominates the head count h and the overhead term $\frac{N^2 d}{h}$ becomes negligible when $h \gg 1$.

For typical configuration values (e.g., SAM (ViT-b) with $d = 768$, $h = 12$):

$$R \approx \frac{1 + \frac{12}{2 \cdot 768} + \frac{2}{12}}{4 + \frac{2 \cdot 12}{768}} \approx 0.29$$

- **Small-N (Windowed Attention):**
  When $N \ll d$:

  $$R_{\text{window}} \approx \frac{\frac{3}{2} N d^2}{3 N d^2} = \frac{1}{2}$$

  where the linear projection term dominates. Since windowed attention already reduces the overall complexity by partitioning the full token sequence into smaller subsets, it transforms the complexity from $O(N^2)$ to $O(k \cdot w^2)$, where N denotes the total number of tokens, k represents the number of windows, and w is the window size. As a result, applying token merging in this context yields relatively moderate but consistent improvements in efficiency.

Generally speaking, since $d' = d/h \ll d$ due to head aggregation, the additional overhead introduced by token merging remains negligible, thereby ensuring that the overall reduction factor $R < 1$ in all scenarios. This analysis confirms that token merging theoretically yields provable efficiency gains across all types of attention blocks, with particularly significant improvements observed in global attention regimes where the token count is much higher.

Layer Selection for Token Merging

After determining the appropriate placement of token merging and unmerging operations within each ViT layer, the subsequent step involves selecting which specific layers should incorporate token merging. It is evident that applying token merging at different depths of the network, whether in shallow or deep layers, can impact the final segmentation performance to varying degrees. Motivated by the pioneering work *ToMe* [3], cosine similarity was adopted as the metric for detecting redundant tokens, with ablation studies that demonstrate its effectiveness over alternative distance measures. Subsequent studies [5, 11, 30, 31, 54, 68, 75] have consistently adopt cosine similarity for recognizing token redundancy.

However, a critical question remains: can cosine similarity reliably identify tokens suitable for merging without compromising segmentation quality? ALGM [54] conducted a detailed investigation into this issue. They analyzed token similarities in Segmenter [71] using the ADE20K [101] training set by calculating average cosine similarities between inter-class and intra-class tokens across all ViT layers. Their findings, illustrated in **Figure 4.7(a)**, revealed that in the early layers, cosine similarity does not effectively reflect semantic correspondence. In contrast, in deeper layers, cosine similarity serves as a more reliable metric for identifying tokens that can be safely merged with minimal impact on segmentation accuracy.

(a) Inter- and intra-token similarity across layers, measured by average cosine similarity on Segmenter (ViT-s) using the ADE20K training set.

(b) Segmentation performance when reducing r = 50% of tokens in the *i*-th layer of SAM (ViT-b) on the DIS5K validation set.

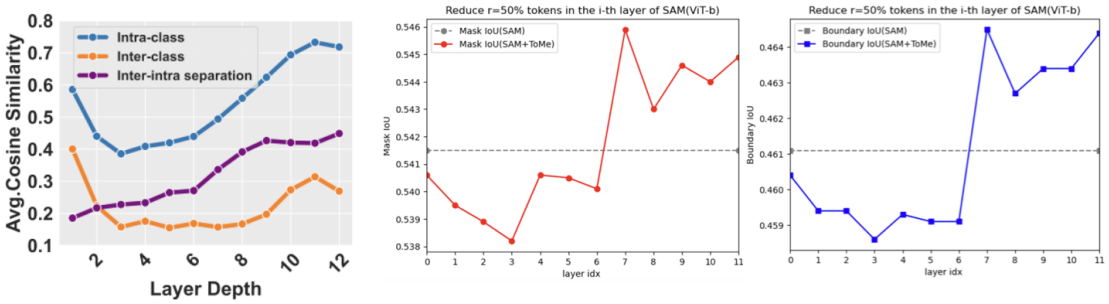


Figure 4.7: Performance Evaluation Per Layer Index.

To validate these findings within our context, we conducted a layer-wise evaluation using SAM (ViT-b) by applying *ToMe* with a fixed reduction rate of $r = 50\%$. The independent variable in our experiment was the layer index, which also determines whether the attention mechanism is windowed or global. To ensure consistency across experiments, we applied *ToMe* with a reduction rate of $r = 50\%$, maintaining fixed token partitions into sets $\mathbb{A}$ and $\mathbb{B}$ across all layers. All tokens in $\mathbb{A}$ were merged with their most similar counterparts in set $\mathbb{B}$, as determined by cosine similarity. The results, shown in **Figure 4.7(b)**, align closely with those of ALGM. Specifically, applying *ToMe* in the initial layers leads to noticeable degradation in segmentation performance, whereas applying it in the deeper layers preserves or even slightly improves accuracy compared to the baseline without token merging.

Based on these insights, we opt to apply token merging exclusively in the last 5 layers of SAM. Each of these layers, when evaluated independently, outperforms the baseline model (**See Figure 4.7(b)**). Furthermore, we explored various combinations of different layers to assess their cumulative impact, with detailed experimental results presented in **Section 4.3.2**.

Selection of Feature Embeddings for Similarity Comparison

To build upon insights from prior studies, the choice of features for token similarity comparison depends both on the target task and the placement of token merging modules within the network. In the context of image classification, as demonstrated in *ToMe* [3], the use of keys has been shown to be the most effective for identifying mergeable tokens. However, in segmentation-focused studies such as ALGM and SegFormer++ [30, 54], token embeddings X are favored. This distinction stems from the differing nature of the two tasks: while image classification relies solely on the final `CLS` token for prediction, semantic segmentation requires the entire set of token embeddings to contribute directly to the output. Consequently, each token holds greater importance in segmentation, making the embeddings a more appropriate choice for token similarity evaluation.

Another crucial factor is the placement of the token merging module within the ViT architecture. When token merging occurs between the MHSA and MLP modules (**Figure 3.7 (a)**), the key representations can be readily accessed from the first linear projection within the MHSA block of the current layer. However, when merging is applied prior to the MHSA operation (**Figure 3.7**

(b)), keys have not yet been computed in this layer, and retrieving them would require backtracking through earlier layers. This is particularly difficult for SAM, as it adopts a periodic windowed-global attention scheme, which means the layer preceding the current one may have different spatial sizes, and merge operations derived from that layer cannot be directly used in the current layer. In such cases, the token embeddings X become the most natural and accessible choice. Given the segmentation task and our specific configuration in which token merging precedes the MHSA module, we likewise adopt token embeddings X as the feature for computing token similarity.

Handling Multi-Head Attention

We also conducted an ablation study on various strategies for aggregating features across attention heads when computing token similarity. Specifically, we applied *PiToMe* using different head aggregation methods on the input token embeddings X, with a fixed reduction rate of $r = 50\%$ applied to the last five layers of SAM. The results, evaluated on the DIS5K validation set, are presented in **Table 4.3**.

Table 4.3: Comparison of different Head Aggregation Strategies for Token Merging.

Method	**Mask IoU (↑)**	**Boundary IoU (↑)**	**(10^9)FLOPs/im (↓)**
SAM (Baseline)	0.5415	0.4611	486.42
+*PiToMe* (Per Head)	0.5205	0.4296	451.61
+*PiToMe* (Concat)	0.5483	0.4663	451.61
+*PiToMe* (Mean)	0.5535	0.4711	425.96

Among the evaluated methods, performing token merging independently for each attention head resulted in a noticeable decline in performance. Prior studies [28, 38, 77] on multi-head self-attention have consistently demonstrated that only a subset of attention heads plays a dominant role in the model's decision-making, while many others contribute minimally. This phenomenon is comparable to an orchestra, where numerous instruments are present, but only a few carry the melody and shape the overall sound. Recognizing this imbalance has inspired research into improving the computational efficiency of ViT models by pruning or disregarding such underutilized heads. In this context, head aggregation can be regarded as a complementary strategy that consolidates information across multiple heads, potentially mitigating the influence of less informative ones while preserving overall model performance.

As shown in **Figure 4.8**, to analyze the variability of token importance across attention heads, we computed energy scores for all token embeddings $\mathbf{X}$ extracted from the final layer of SAM (ViT-B). For each attention head $h \in \{0, 1, \ldots, 11\}$, we calculated the following statistics: **(i) Range**: $\text{Range}^h(\text{EnergyScore}^h(\mathbf{X})) = \max(\text{EnergyScore}^h(\mathbf{X})) - \min(\text{EnergyScore}^h(\mathbf{X}))$ and **(ii) Mean Absolute Deviation (MAD)**: $\text{MAD}^h(\text{EnergyScore}^h(\mathbf{X})) = \frac{1}{N}\sum_{i=1}^{N}\left|\text{EnergyScore}^h(\mathbf{x}_i) - \mu^h\right|$, where $\mu^h = \frac{1}{N}\sum_{i=1}^{N}\text{EnergyScore}^h(\mathbf{x}_i)$ is the mean energy score over all tokens $\mathbf{x}_i \in \mathbf{X}$, and N denotes the total number of tokens. These statistics were computed for each input image in the DIS5K validation set. The resulting distributions of range and mean absolute deviation values across heads were subsequently visualized. As illustrated in the two box plots, our experiment demonstrates that the energy scores, which are used to assess the importance of tokens, exhibited

minimal variation across most attention heads, while noticeable differences appeared only in a few dominant ones (head_5 and head_7). Since *PiToMe* relies on these scores to suppress low similarity values and to emphasize pairs of tokens that are strongly similar, whose goal is to increase the margin that distinguishes tokens suitable for merging from those that are not. However, because underutilized heads produce relatively uniform energy scores, they are less effective at identifying informative distinctions between tokens. As a result, performing token merging independently for each head may introduce noise and ultimately degrade the quality of the merging process.

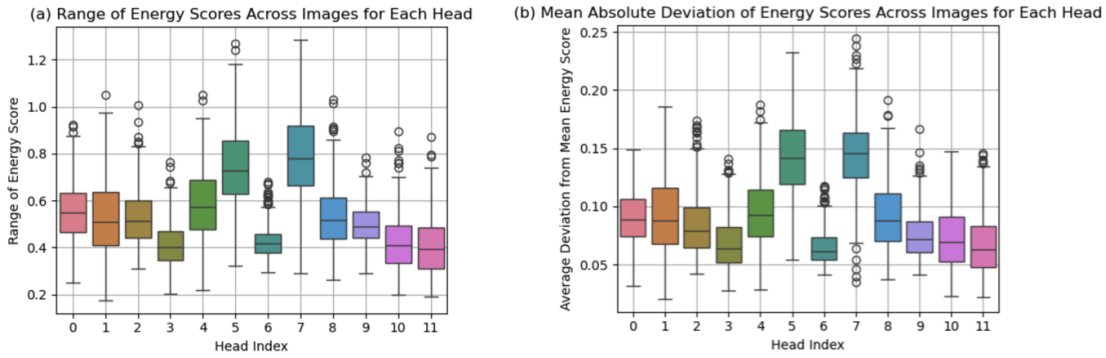


Figure 4.8: Per-Head Variability in Energy Scores.

To address this issue, we investigated two commonly used strategies for aggregating head-wise features: concatenation and mean pooling. Mean aggregation reduces the embedding dimensionality, thereby enhancing computational efficiency and simplifying the representation. Interestingly, in contrast to the results reported in *ToMe* [3], where concatenation outperformed mean pooling for classification tasks, our findings demonstrate that mean aggregation yields better performance in segmentation tasks, with both aggregation strategies surpassing the baseline of the baseline SAM model without token merging. We hypothesize that this discrepancy stems from the differing nature of the tasks. Classification tasks require the capture of high-dimensional and fine-grained interactions between tokens, with the `[CLS]` token ultimately serving as the central feature. In contrast, segmentation tasks benefit from spatial coherence and more compact representations across the entire token set. The reduced dimensionality resulting from mean aggregation may enhance the stability of similarity comparisons, leading to more effective token merging decisions in dense prediction tasks such as segmentation.

Adapting Decomposed Relative Positional Embeddings (DRPE)

As discussed in **Section 4.1.2.2**, the Multi-Head Self-Attention (MHSA) mechanism in SAM incorporates decomposed relative positional embeddings (DRPE), which enhance spatial sensitivity but introduce additional challenges when integrating token merging.

In standard MHSA without relative positional encoding, attention computation depends solely on pairwise interactions among tokens. As such, after merging, the order or spatial arrangement of the reduced token sequence does not need to be preserved. In the initial design of *ToMe*, which was developed for plain ViT models, the reduced token sequence was formed by concatenating the set of unmerged tokens with the destination tokens, each updated by averaging with its corresponding

source tokens. However, with DRPE, relative spatial relationships between tokens become critical, as the DRPE vectors are added to each query-key pair based on their original positions. This necessitates careful handling of positional information post-merging to ensure the correct positional bias is applied.

To address this, we explicitly track the original spatial positions of all retained tokens after the merging process. Unmerged tokens remain unaffected and preserve their original positions. For merged tokens, specifically the destination tokens into which source tokens are merged, we assume, based on the fundamental principle of token merging, that the source and destination tokens are sufficiently similar. Consequently, the original position of the destination token is adopted to represent the merged entity within the reduced token set.

Let $\mathcal{T} = \{t_1, \ldots, t_N\}$ denote the original token set with corresponding positions $\mathcal{P} = \{p_1, \ldots, p_N\}$. After merging, the positions of reduced token set $\mathcal{P}' \subseteq \mathcal{P}$ retains:

- Unmerged tokens, each preserving its original position from $\mathcal{P}$,
- Destination tokens, each assigned its original position, now representing a group of merged tokens.

These positions are used to index into the learned DRPE embedding table, ensuring that the correct relative positional embeddings are applied to each query-key pair in the reduced attention map. This approach effectively preserves spatial coherence and maintains attention accuracy even after token reduction. Its simplicity ensures seamless integration without disrupting SAM's core architectural mechanisms.

4.3.2 Experimental Results and Discussion

In this section, we present the main experimental results of integrating token merging algorithms, which are introduced in **Section 3.3.2**, into SAM (ViT-B). Our objective is to evaluate how token merging can reduce computational costs while preserving segmentation performance. We investigate the performance trade-offs associated with various merging strategies and configurations in high-resolution image segmentation tasks.

To provide a comprehensive analysis, we examine the results from three perspectives. Firstly, we investigate how the selection of layers for applying token merging influences model performance. Secondly, we conduct an analysis of the key factors that drive its influence on segmentation accuracy. Finally, we compare the effectiveness of different token merging algorithms in achieving an optimal balance between segmentation quality and computational efficiency. The following subsections present detailed findings corresponding to each of these aspects.

4.3.2.1 Impact of Token Merging Layer Placement

We evaluated three layer configurations for applying token merging: **(i)** merging $r = 50\%$ of tokens in the last five layers (**Table 4.4** and **Figure 4.9**), **(ii)** merging $r = 50\%$ of tokens in the layers that perform global attention—specifically the 2nd, 5th, 8th, and 11th layers (**Table 4.5** and **Figure 4.10**), and **(iii)** merging $r = 50\%$ of tokens across all layers (**Table 4.6** and **Figure 4.11**).

These configurations were selected based on the following motivations. For **(i)**, our single-layer ablation study (see **Figure 4.7(b)**) indicates that applying token merging to any of the last five layers improves both `Mask IoU` and `Boundary IoU`, whereas merging in earlier stage consistently degrades performance. This suggests that limiting token merging to the later layers may preserve or even enhance segmentation accuracy. For **(ii)**, since SAM employs a periodic windowed-global attention pattern to introduce *inductive bias* and improve efficiency, only a few layers perform global attention. Merging tokens at these layers allows us to assess how reducing tokens in globally-attentive layers influences segmentation performance. For **(iii)**, merging tokens in all layers represents the most aggressive reduction strategy and serves to evaluate the extent to which computational savings can be achieved, as well as the degree to which segmentation accuracy may be compromised.

(i) Merge $r = 50\%$ tokens in the last five layers of SAM(ViT-b).

Although the computational savings are moderate, with FLOPs per image reduced by slightly over 10%, the segmentation performance is largely preserved and, in many cases, even improved. Specifically, *ToMe* achieves a slight average performance gain, notably improving results on DIS and HRSOD datasets, while incurring only minor performance degradation on COIFT. *ToMeSD* variants show mixed results, with some configurations generating gains in ThinObject and HRSOD, although with slight decreases in DIS. Among all tested variants, *PiToMe* stands out by achieving the highest average performance improvement, including a notable increase of over 1% in mask IoU and boundary IoU on the DIS dataset, despite having no additional architectural modifications or tuning. These findings confirm our hypothesis that applying token merging in the final layers, where feature representations are more semantically rich, can preserve, or even enhance segmentation performance, while still contributing to computational efficiency.

Model	**DIS**		**COIFT**		**HRSOD**		**ThinObject**		**Average**		(10^9)
	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	**FLOPs/im↓**
SAM (Baseline)	0.5415	0.4611	0.8775	0.8204	0.8645	0.7816	0.5439	0.4605	0.7069	0.6309	486.42
+ToMe	0.5464 ↑0.0049	0.4632 ↑0.0021	0.8715 ↓0.0060	0.8160 ↓0.0044	0.8683 ↑0.0038	0.7853 ↑0.0037	0.5463 ↑0.0024	0.4601 ↓0.0004	0.7081 ↑0.0012	0.6312 ↑0.0003	424.21 ↓13%
+ToMeSD(Rand 25%)	0.5395 ↓0.0020	0.4585 ↓0.0026	0.8749 ↓0.0026	0.8193 ↓0.0011	0.8667 ↑0.0022	0.7840 ↑0.0024	0.5496 ↑0.0057	0.4646 ↑0.0041	0.7077 ↑0.0008	0.6316 ↑0.0007	424.07 ↓13%
+ToMeSD(Stride 2×2)	0.5354 ↓0.0061	0.4537 ↓0.0074	0.8726 ↓0.0049	0.8174 ↓0.0030	0.8687 ↑0.0042	0.7856 ↑0.0040	0.5456 ↑0.0017	0.4600 ↓0.0005	0.7056 ↓0.0013	0.6292 ↓0.0017	424.07 ↓13%
+ToMeSD(Rand 2×2)	0.5366 ↓0.0049	0.4554 ↓0.0057	0.8752 ↓0.0023	0.8199 ↓0.0005	0.8681 ↑0.0036	0.7861 ↑0.0045	0.5485 ↑0.0046	0.4632 ↑0.0027	0.7071 ↑0.0002	0.6312 ↑0.0003	424.07 ↓13%
+PiToMe	0.5535 ↑0.0120	0.4711 ↑0.0100	0.8761 ↓0.0014	0.8173 ↓0.0031	0.8692 ↑0.0047	0.7892 ↑0.0076	0.5455 ↑0.0016	0.4618 ↑0.0013	0.7111 ↑0.0042	0.6349 ↑0.0040	425.96 ↓12%

Table 4.4: Merge $r = 50\%$ tokens in the last five layers of SAM(ViT-b).

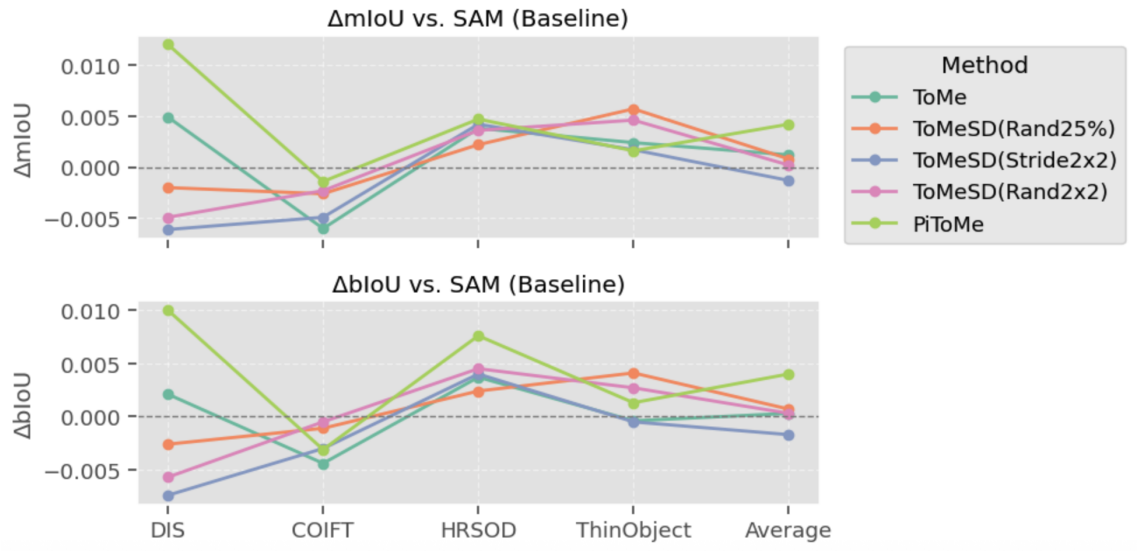


Figure 4.9: Performance ranking after merging $r = 50\%$ tokens in the last five layers of SAM(ViT-b).

(ii) Merge $r = 50\%$ tokens in layers which do global attention of SAM(ViT-b).

This configuration achieves an approximately 20% reduction in FLOPs per image but generally results in decreased segmentation performance relative to the baseline. The performance drop is minor for *PiToMe* and *ToMe*, with slight improvements observed on some datasets, whereas *ToMeSD* shows a more pronounced average decline exceeding 1%. These findings suggest that applying token merging to globally attended layers is harmful to model performance. In SAM's architecture, only a limited number of layers employ global attention, making them essential for capturing long-range dependencies and contextual information. Disrupting these layers through token merging likely impairs the model's ability to effectively integrate global context, thereby reducing segmentation accuracy.

Model	**DIS**		**COIFT**		**HRSOD**		**ThinObject**		**Average**		(10^9)
	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	**FLOPs/im↓**
SAM (Baseline)	0.5415	0.4611	0.8775	0.8204	0.8645	0.7816	0.5439	0.4605	0.7069	0.6309	486.42
+ToMe	0.5387 ↓0.0028	0.4563 ↓0.0048	0.8708 ↓0.0067	0.8150 ↓0.0054	0.8686 ↑0.0041	0.7849 ↑0.0033	0.5375 ↓0.0064	0.4536 ↓0.0069	0.7039 ↓0.0030	0.6275 ↓0.0034	394.88 ↓19%
+ToMeSD(Rand 25%)	0.5208 ↓0.0207	0.4420 ↓0.0191	0.8714 ↓0.0061	0.8135 ↓0.0069	0.8579 ↓0.0066	0.7740 ↓0.0076	0.5315 ↓0.0124	0.4516 ↓0.0089	0.6954 ↓0.0115	0.6202 ↓0.0107	394.61 ↓19%
+ToMeSD(Stride 2×2)	0.5201 ↓0.0214	0.4404 ↓0.0207	0.8713 ↓0.0062	0.8120 ↓0.0084	0.8611 ↓0.0034	0.7781 ↓0.0035	0.5291 ↓0.0148	0.4493 ↓0.0112	0.6954 ↓0.0115	0.6200 ↓0.0109	394.61 ↓19%
+ToMeSD(Rand 2×2)	0.5208 ↓0.0207	0.4414 ↓0.0197	0.8724 ↓0.0051	0.8140 ↓0.0064	0.8599 ↓0.0046	0.7755 ↓0.0061	0.5315 ↓0.0124	0.4505 ↓0.0100	0.6962 ↓0.0107	0.6204 ↓0.0105	394.61 ↓19%
+PiToMe	0.5421 ↑0.0006	0.4621 ↑0.0010	0.8749 ↓0.0026	0.8185 ↓0.0019	0.8660 ↑0.0015	0.7827 ↑0.0011	0.5417 ↓0.0022	0.4578 ↓0.0027	0.7062 ↓0.0007	0.6303 ↓0.0006	398.10 ↓18%

Table 4.5: Merge $r = 50\%$ tokens in layers which do global attention of SAM(ViT-b).

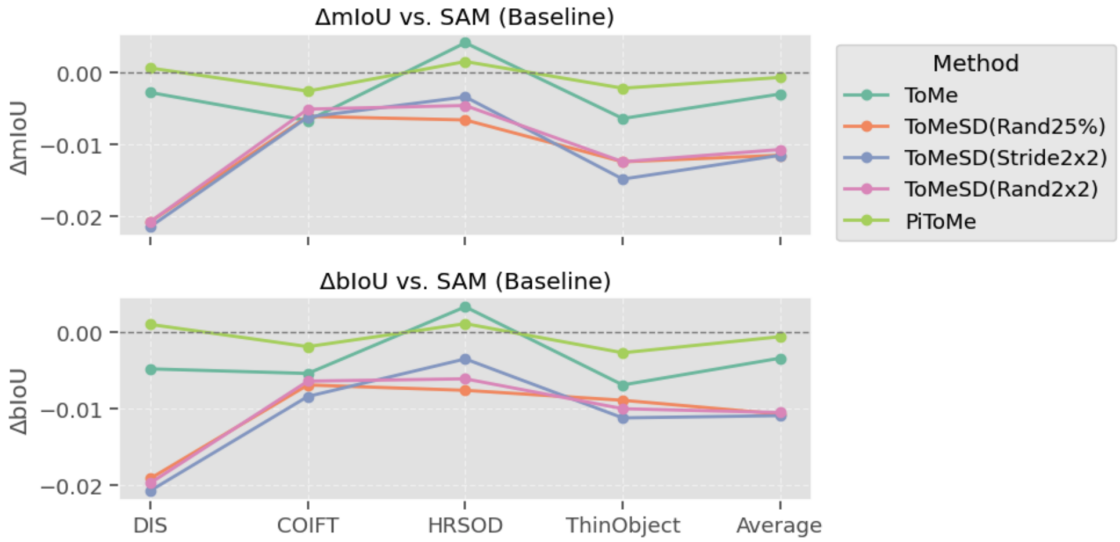


Figure 4.10: Performance ranking after merging $r = 50\%$ tokens in layers which do global attention of SAM(ViT-b).

(iii) Merge $r = 50\%$ tokens in all layers of SAM(ViT-b).

This configuration yields a substantial computational gain, reducing FLOPs per image by nearly 30%. However, this benefit comes at the cost of a notable decline in segmentation performance. All token merging methods exhibit performance degradation in both Mask IoU and Boundary IoU across all datasets. *ToMeSD* variants show the most significant drop, with an average decrease of 2–3%, highlighting their sensitivity to aggressive merging strategies. In contrast, *PiToMe* achieves a more balanced trade-off, exhibiting a relatively modest average decline of 0.53%. These results indicate that applying token merging uniformly across all layers can excessively impair the model's ability to preserve fine-grained and global information, ultimately leading to greater degradation in segmentation quality despite the greater gains in computational efficiency.

Model	**DIS**		**COIFT**		**HRSOD**		**ThinObject**		**Average**		(10^9)
	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	**FLOPs/im↓**
SAM (Baseline)	0.5415	0.4611	0.8775	0.8204	0.8645	0.7816	0.5439	0.4605	0.7069	0.6309	486.42
+ToMe	0.5189 ↓0.0226	0.4328 ↓0.0283	0.8534 ↓0.0241	0.7880 ↓0.0324	0.8524 ↓0.0121	0.7649 ↓0.0167	0.5182 ↓0.0257	0.4306 ↓0.0299	0.6857 ↓0.0212	0.6041 ↓0.0268	351.05 ↓28%
+ToMeSD(Rand 25%)	0.5121 ↓0.0294	0.4307 ↓0.0304	0.8608 ↓0.0167	0.8012 ↓0.0192	0.8468 ↓0.0177	0.7628 ↓0.0188	0.5199 ↓0.0240	0.4388 ↓0.0217	0.6849 ↓0.0220	0.6084 ↓0.0225	350.75 ↓28%
+ToMeSD(Stride 2×2)	0.5017 ↓0.0398	0.4209 ↓0.0402	0.8575 ↓0.0200	0.7961 ↓0.0243	0.8416 ↓0.0229	0.7588 ↓0.0228	0.5044 ↓0.0395	0.4278 ↓0.0327	0.6763 ↓0.0306	0.6009 ↓0.0300	350.75 ↓28%
+ToMeSD(Rand 2×2)	0.5131 ↓0.0284	0.4316 ↓0.0295	0.8638 ↓0.0137	0.8048 ↓0.0156	0.8469 ↓0.0176	0.7634 ↓0.0182	0.5213 ↓0.0226	0.4396 ↓0.0209	0.6863 ↓0.0206	0.6099 ↓0.0210	350.75 ↓28%
+PiToMe	0.5403 ↓0.0012	0.4541 ↓0.0070	0.8674 ↓0.0101	0.8038 ↓0.0166	0.8619 ↓0.0026	0.7750 ↓0.0066	0.5369 ↓0.0070	0.4485 ↓0.0120	0.7016 ↓0.0053	0.6203 ↓0.0106	354.64 ↓27%

Table 4.6: Merge $r = 50\%$ tokens in all layers of SAM(ViT-b).

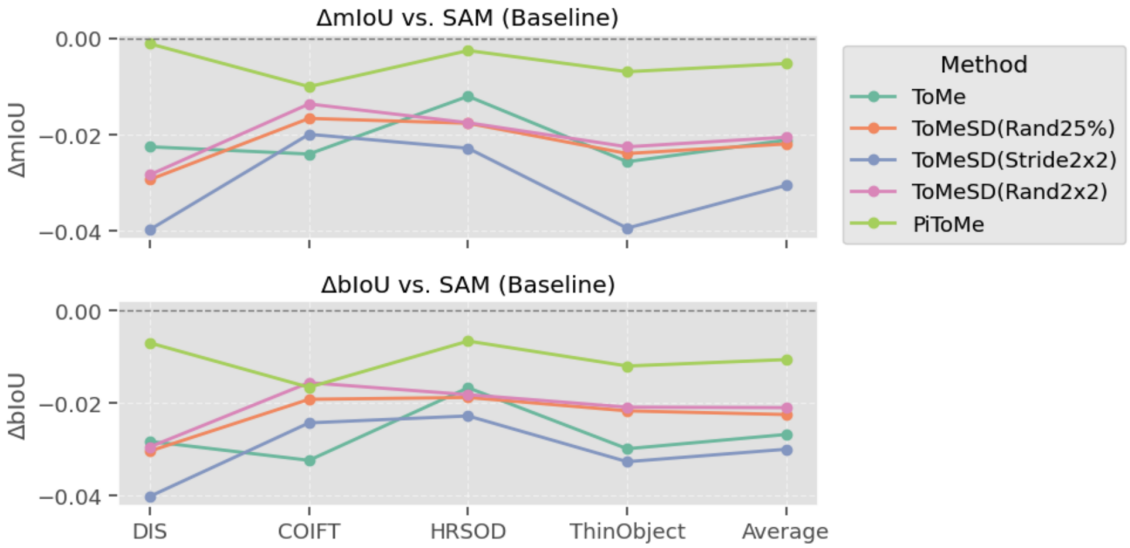


Figure 4.11: Performance ranking after merging $r = 50\%$ tokens in all layers of SAM(ViT-b).

These results collectively highlight the importance of strategic token merging placement within SAM. Different layers serve distinct roles in the representation learning process, capturing varying levels of semantic and spatial information. As demonstrated across the three settings, indiscriminate or overly aggressive merging, particularly in early or globally attentive layers, can significantly degrade segmentation quality despite offering computational savings. Conversely, selective merging in carefully chosen layers can preserve performance while still reducing FLOPs. This underscores the need for thoughtful design in token merging strategies. Meanwhile, as discussed in **Section 4.3.1**, the same merging metric exhibits varying degrees of effectiveness depending on the specific layer, further emphasizing the importance of layer-wise analysis in designing efficient models. Therefore, placement of which layers is not only a technical choice but also a critical factor in balancing efficiency and performance.

4.3.2.2 Reasoning of Segmentation Improvement

From the experimental results of merging 50% of tokens in the last five layers (see **Table 4.4**), it is noteworthy that applying token merging, without any modifications to the model architecture or retraining, not only improves computational efficiency but also enhances segmentation accuracy. This unexpected gain in performance motivates a deeper investigation into the underlying factors contributing to this improvement. In this section, we analyze the possible reasons behind this phenomenon from four perspectives: **(i)** the aggressive reduction rate r in segmentation tasks, **(ii)** the cumulative advantages gained from applying token merging and unmerging across successive layers, **(iii)** the denoising effect of token merging, and **(iv)** the balancing of spatial feature representations.

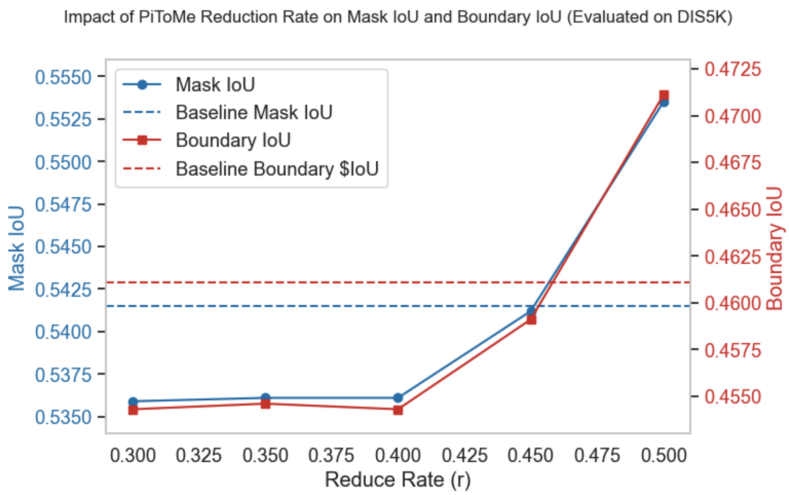


Figure 4.12: Impact of Increasing Reduction Rate r on Segmentation Performance.

Aggressive Reduction Rate r Previous studies such as [3, 5, 30, 75] on token reduction for improving the efficiency of Vision Transformers, commonly report a trade-off between performance and computational savings. As the reduction rate r increases, performance typically degrades due to information loss introduced by token merging. These methods generally tend to favor merging only the most similar token pairs or employ relatively high similarity thresholds, ensuring that merging occurs predominantly among highly redundant tokens in order to preserve segmentation quality. However, in our experiments applying *PiToMe* to the last five layers, we observe a different trend. As shown in **Figure 4.12**, segmentation performance remains relatively stable for r between 0.3 and 0.4, slightly below the baseline. Interestingly, as r increases to 0.45 and 0.5, the model begins to outperform the baseline, with over 1% improvement in IoU metrics eventually.

We hypothesize that this improvement is attributable to the unique nature of segmentation tasks. Unlike classification, which relies on holistic representations of entire images, segmentation emphasizes fine-grained spatial details, especially object boundaries, where local contrast is crucial. Token merging operates greedily by fusing the most similar token pairs. At higher reduction rates, less similar tokens are forced to merge, which may act as a form of *regularization*. Specifically, this process can enhance feature representations by reinforcing boundary distinctions when these less similar tokens are later unmerged via copying averaged feature embedding vectors. Consequently, aggressive merging may help amplify important differences across regions, particularly near boundaries in complex datasets such as DIS5K, leading to improved segmentation accuracy.

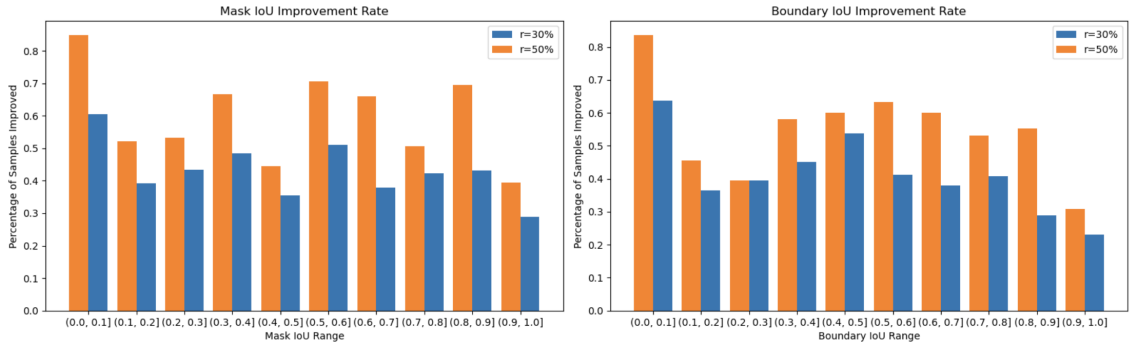


Figure 4.13: Percentage of Samples That Show Improvement Under Different Reduction Rates r.

In order to examine whether the observed improvements can be generalized across most inputs and whether they depend on the baseline performance, i.e., the quality of feature representations learned by the model. We visualize, under different reduction rates r, the proportion of samples that exhibit improved segmentation performance based on their baseline range in **Figure 4.13**. It is evident that when $r = 0.5$, a larger number of image inputs, well more than half, achieve improved results compared to using $r = 0.3$. A qualitative comparison is presented in **Figure 4.17**, where the segmentation outputs are shown for representative examples. In each image, the red contour delineates the boundary of the predicted mask, while the green contour represents the ground truth. The overlapping areas and contours illustrate the alignment between the prediction and the ground truth. These examples demonstrate that more aggressive merging can lead to improved segmentation, particularly for tokens within complex regions or along intricate boundaries. These quantitative and qualitative results further support the claim that a more aggressive merge rate can enhance segmentation quality by unmerging and substituting suboptimal segmentations learned by the model with values from a reasonably matched and more reliable averaged segment.

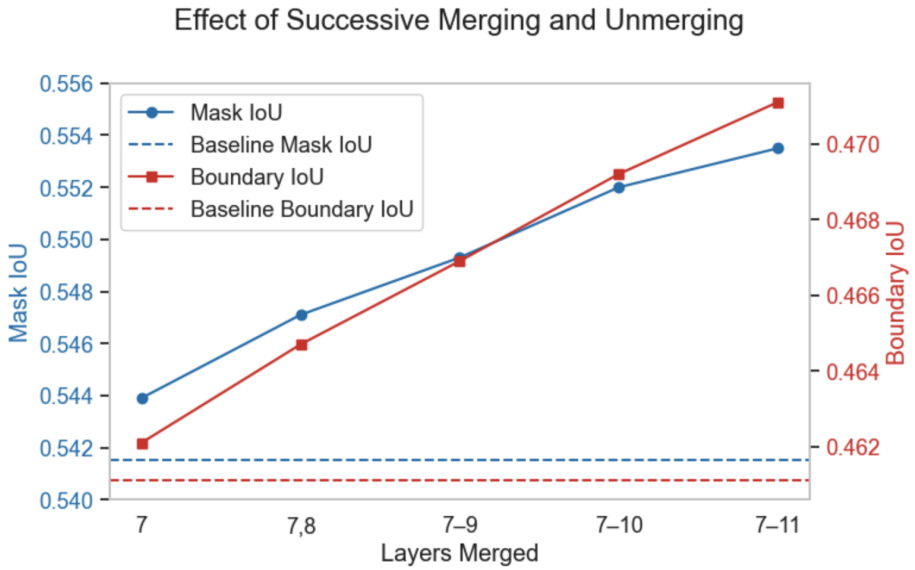


Figure 4.14: Successive Token Merging and Unmerging with *PiToMe* in last 5 layers of SAM on DIS5K Val Set.

Progressive Gains Through Successive Merging and Unmerging The results illustrated in **Figure 4.14** clearly demonstrate that applying *PiToMe* progressively across the deeper layers of the SAM encoder leads to consistent improvements in both mask IoU and boundary IoU. As token merging and unmerging are successively applied from the 7th to the 11th layer, the performance exhibits a steady upward trend. This suggests that the process of merging via feature averaging, followed by unmerging, accumulates beneficial effects on feature representation in the later stages of the encoder. Since deeper layers encode more semantically rich and aggregated information, the merging process becomes increasingly effective at refining token representations step by step. Qualitative comparisons illustrating the improvement in segmentation quality with successive merging are presented in **Figure 4.18**. Although the gains per layer are moderate (0.2% to 0.3%), the cumulative improvements notably exceed the baseline, thereby underscoring the importance of strategically selecting layers for token merging to maximize its overall effectiveness.

Denoising & Balancing Token merging can contribute to both denoising and balancing effects, as previously hypothesized in ALGM [54], which explored token merging in plain ViT models for semantic segmentation. In this context, **denoising** refers to the process of averaging token values during merging, which smooths noisy representations and potentially facilitates learning. **Balancing** occurs when tokens corresponding to frequently appearing or large-area categories are merged, reducing their dominance in the self-attention mechanism. This leads to a more balanced attention distribution that benefits rarer classes, particularly since token merging is applied only within the self-attention modules of SAM in our cases.

To investigate these effects, we conducted ablation experiments using *PiToMe* with a reduction rate of $r = 50\%$ in the last five layers, tested on the DIS dataset, a configuration previously shown to yield optimal segmentation performance. For **denoising**, we compared averaging all merged token pairs against simply selecting the destination tokens. For **balancing**, we evaluated a variant where merged tokens were not reduced in number but replaced by replicated averaged embeddings, thereby eliminating the balancing effect in the attention computation.

The results, summarized in **Table 4.7**, show that disabling either denoising or balancing leads to decreased performance, and disabling both results in the most significant degradation. These findings provide empirical support for the beneficial roles of denoising and balancing in improving segmentation quality through token merging.

Table 4.7: Segmentation Performance Improvement Analysis (Denoising & Balancing).

Denoising	Balancing	Token selection	Token reorganization	mIoU↑
✓	✓	Take average	Reduction	0.5535
✓	✗	Take average	No reduction	0.5471 ↓0.0064
✗	✓	Pick dst token	Reduction	0.5489 ↓0.0046
✗	✗	Pick dst token	No reduction	0.5454 ↓0.0081

4.3.2.3 Impact of Token Merging Algorithms

As demonstrated by the experimental results presented in **Table 4.4**, **Figure 4.9**, **Table 4.5**, **Figure 4.10**, **Table 4.6** and **Figure 4.11**, different token merging algorithms exhibit varying effects on both segmentation performance and computational efficiency when integrated into different layers of SAM (ViT-B). This section analyzes the underlying factors contributing to these differences.

Computational Efficiency The empirical FLOPs per image analysis align consistently with the theoretical computational complexity analysis in **Section 4.3.1** and **Table 4.2**. Among the methods, *PiToMe* incurs the highest computational cost due to its reliance on a fully connected token graph. This requires computing pairwise cosine similarities among all tokens to derive energy scores for partitioning, although the resulting similarity matrix can later be reused during the greedy matching phase. In contrast, *ToMe* and *ToMeSD* perform partitioning soley based on spatial grids. After dividing tokens into two disjoint sets, $\mathbb{A}$ and $\mathbb{B}$, similarity is computed only between tokens across these sets, resulting in a lower computational burden compared to *PiToMe*. Furthermore, *ToMeSD*, which is designed to support more aggressive merging, tends to allocate a larger set $\mathbb{A}$ and a smaller set $\mathbb{B}$, requiring fewer similarity computations than the evenly split sets used in *ToMe*. In general, due to head aggregation and the shared use of *Bipartite Soft Matching*, the computational overhead of running the SAM image encoder across these algorithms remains roughly comparable when the same reduction rate r is applied.

Effectiveness Experimental results clearly demonstrate that incorporating semantic information into token partitioning significantly enhances the quality of token merging. Specifically, it increases the likelihood that tokens in set $\mathbb{A}$ can find highly similar counterparts in set $\mathbb{B}$, ensuring that the resulting merged pairs are semantically coherent. Among the token merging algorithms evaluated, *PiToMe* consistently outperforms others in preserving segmentation accuracy. This advantage becomes particularly evident under aggressive configurations, such as when merging is applied across all layers utilizing global attention (**Table 4.5** and **Figure 4.10**), or even across all layers in the network (**Table 4.6** and **Figure 4.11**). In these settings, methods including *ToMe* and *ToMeSD* often suffer from noticeable performance degradation (1% to 3% drop in IoU), whereas *PiToMe* maintains performance levels that are much closer to the SAM baseline, which does not incorporate any merging.

A key design contributing to *PiToMe*'s effectiveness is its use of energy scores for guiding token partitioning. The energy score, derived from the redundancy of tokens' feature representation, provides an estimate of how many other tokens are similar to a given token. Tokens with high energy scores are surrounded by many similar tokens and are thus more suitable candidates for merging, while those with low energy scores tend to be more unique or informative and should be preserved. Tokens are sorted by their energy scores and then alternately assigned to sets $\mathbb{A}$ and $\mathbb{B}$, which increases the likelihood that similar tokens are placed into opposite sets and can thus be matched later. This strategy is grounded in the assumption that similar tokens tend to have similar energy scores and will appear near each other in the sorted list. By introducing this energy-based partitioning, *PiToMe* enhances the semantic coherence of merging, even when protective mechanism, which is designed to exclude low-energy (i.e., informative) tokens, are not activated, as is the case when employing the maximum reduction rate (e.g., $r = 0.5$). Despite the

absence of these protections during aggressive merging, the underlying energy-based partitioning continues to contribute significantly to the method's robustness, as illustrated in **Figure 4.19** through a qualitative comparison.

In contrast, grid-based methods such as *ToMe* and *ToMeSD* do not account for semantic information during partitioning. Instead, they divide tokens spatially, which may not align well with semantic boundaries. Between these two, *ToMe* generally preserves segmentation performance better than *ToMeSD*, likely due to the larger destination set $\mathbb{B}$ used during merging. A larger $\mathbb{B}$ increases the probability that each token in the source set $\mathbb{A}$ can find a well-matched token to merge with, thus maintaining segmentation quality more effectively. *ToMeSD*, in contrast, uses a smaller $\mathbb{B}$ to enable more aggressive reduction, which saves computation but can lead to suboptimal matches and poorer performance. Among the *ToMeSD* variants, the **Rand 2×2** configuration, which combines stride-based grid sampling with irregularity, outperforms other variants including **Rand 25%** (lacks grid alignment) and **Stride 2×2** (lacks randomness). This result suggests that introducing controlled randomness into the grid sampling helps avoid rigid token grouping and increases match flexibility.

These findings underscore a trade-off between effectiveness and efficiency in token partitioning. A smaller destination set $\mathbb{B}$ improves computational efficiency by reducing the number of similarity computations and raising the theoretical upper bound of the reduction rate r, thereby decreasing the number of tokens passed to the attention blocks. In contrast, a larger and more representative $\mathbb{B}$ enhances effectiveness by improving the quality of token matching but imposes greater limitations on the number of tokens that can be merged. Ideally, a token partition strategy for a BSM-based token merging process should aim for a reduced yet representative destination set $\mathbb{B}$ that preserves semantic proximity to maintain segmentation performance under aggressive merging.

4.4 Further improvements: *GradToMe*

In the previous section, we demonstrated that context-aware token partitioning, such as the energy score in *PiToMe*, is effective in dividing tokens into sets $\mathbb{A}$ and $\mathbb{B}$ in a way that increases the likelihood of forming semantically meaningful matched pairs. This strategy helps preserve model performance after token merging. Building on the strengths of *PiToMe*, this section identifies its key limitations and introduces an improved token merging algorithm, *GradToMe*, specifically designed to address these shortcomings. We then evaluate its effectiveness in preserving the spectral characteristics of token representations, as achieved by *PiToMe*.

4.4.1 I: Replacing the Energy Score

In *PiToMe*, the energy score is computed by aggregating all pairwise cosine similarities, shifted by a margin, to estimate how redundant or informative each token is. This metric effectively captures how many tokens are similar to a given one and is further processed with an ELU activation to emphasize tokens within large, highly redundant clusters (see **Section 3.3.2.2** for a detailed formulation). However, this computation involves a quadratic complexity of $O(N^2)$ with respect to the number of input tokens N, as it requires evaluating all pairwise similarities.

Motivated by the need for a more efficient approach to estimate token importance while retaining contextual awareness, we explore replacing the energy score with a gradient-based approximation inspired by classical image processing. In this context, gradient magnitude is often used to highlight informative structures: tokens with higher gradients are likely to represent important local variations, whereas lower gradients suggest redundancy or smooth regions. This intuition makes gradient-based measures a natural candidate for identifying salient tokens. Accordingly, the informative tokens requiring protection previously identified by low energy scores are now determined by high gradient magnitudes. We begin with simple first-order gradient approximations, implementing two widely used operators—Central Differences and the Sobel Operator. These methods offer computational advantages: central differences require only the four immediate neighbors of each token, and the Sobel operator considers a 3×3 neighborhood (see **Section 3.4** for details). Both methods scale linearly with the number of tokens, making them potentially more efficient alternatives for guiding token partitioning in the merging process.

Computational Efficiency **Table 4.8** presents the efficiency improvements achieved by specifically evaluating the FLOP counts of the token merging block for a given input processed through either global or window attention. The results demonstrate a substantial gain in efficiency when replacing the energy score with either form of first-order gradient approximation, regardless of the type of attention block. This substitution leads to a reduction in computational complexity, transitioning from quadratic to linear with respect to the number of tokens.

Attention	**Method**	(10^9) **FLOPs/im**↓
Global	PiToMe	1.0737
	GradToMe(Central Diff)	0.2684 ↓75.00%
	GradToMe(Sobel)	0.2732 ↓74.56%
Window	PiToMe	0.0615
	GradToMe(Central Diff)	0.0154 ↓74.96%
	GradToMe(Sobel)	0.0210 ↓65.85%

Table 4.8: FLOP counts analysis specific to different token merging methods considering inputs passing global or window attention.

Segmentation Performance In **Table 4.9** and **Figure 4.15**, we present experimental results in which the energy score of *PiToMe* is replaced by two alternative first-order gradient approximation methods. The token merging configuration remains consistent, being applied to the final five layers of the model. The average difference between gradient-based approximations and the original energy score is negligible—less than 0.1% in both Mask IoU and Boundary IoU. This margin is much smaller than the performance gap between context-unaware merging methods, further reinforcing the efficacy of the gradient-based approach. Performance differences across individual datasets are mixed: in some cases, gradient approximations outperform the energy score, while in others, the energy-based method performs slightly better. However, these variations are minimal and fall within the range of fluctuation caused by changes in floating-point precision or random seeds. Between the two gradient approximation techniques, central differences yield an average

improvement of 0.05% in Mask IoU and 0.03% in Boundary IoU. However, these differences are statistically insignificant and inconsistent across datasets. Consequently, we adopt the Sobel operator as the default method for computing gradient magnitudes, not only because it considers a larger neighborhood than central differences, thereby broadening its perceptual scope, but also due to its use of a weighted kernel, which theoretically offers greater robustness to noise (discussed in **Section 3.4**).

Model	DIS		COIFT		HRSOD		ThinObject		Average	
	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑
SAM (Baseline)	0.5415	0.4611	0.8775	0.8204	0.8645	0.7816	0.5439	0.4605	0.7069	0.6309
SAM+PiToMe	0.5535 ↑0.0120	0.4711 ↑0.0100	0.8761 ↓0.0014	0.8173 ↓0.0031	0.8692 ↑0.0047	0.7892 ↑0.0076	0.5455 ↑0.0016	0.4618 ↑0.0013	0.7111 ↑0.0042	0.6349 ↑0.0040
SAM+GradToMe(Central Diff)	0.5520 ↑0.0105	0.4695 ↑0.0084	0.8764 ↓0.0011	0.8192 ↓0.0012	0.8688 ↑0.0043	0.7883 ↑0.0067	0.5470 ↑0.0031	0.4624 ↑0.0019	0.7110 ↑0.0041	0.6349 ↑0.0040
SAM+GradToMe(Sobel)	0.5538 ↑0.0123	0.4714 ↑0.0103	0.8748 ↓0.0027	0.8179 ↓0.0025	0.8692 ↑0.0047	0.7891 ↑0.0075	0.5441 ↑0.0002	0.4599 ↓0.0006	0.7105 ↑0.0036	0.6346 ↑0.0037

Table 4.9: Segmentation performance when replacing the energy score with first-order gradient approximations.

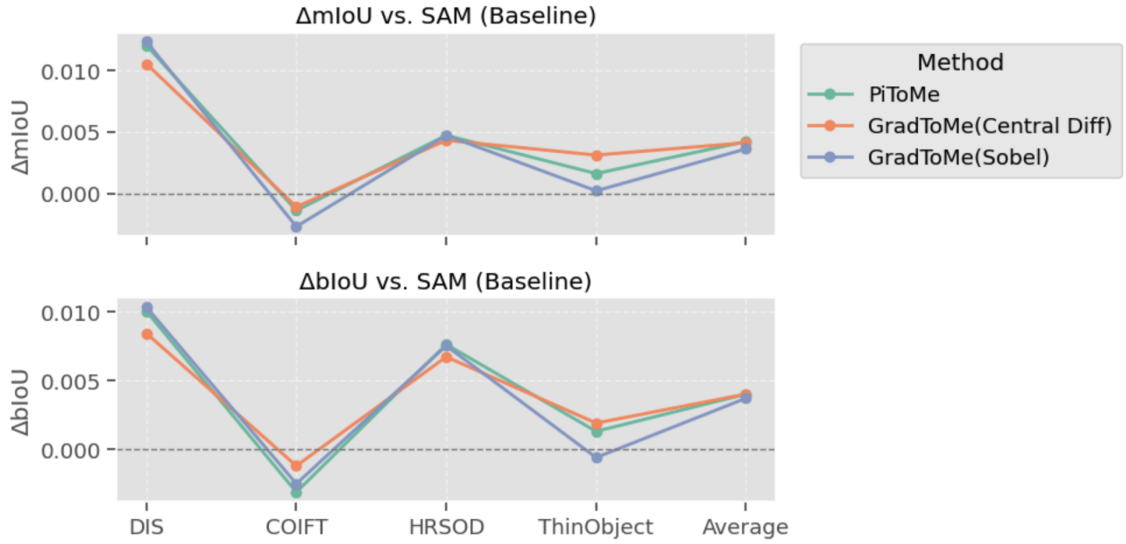


Figure 4.15: Performance ranking when replacing the energy score with first-order gradient approximations.

Correlation Analysis In addition to directly evaluating segmentation performance on benchmark datasets to assess whether gradient approximation can serve as a viable alternative to the energy score, we also conducted a statistical analysis. Gradient approximation kernels, such as the Sobel operator, were originally designed for images with feature representations encoded as RGB values—a transparent and interpretable domain. The Sobel operator, in particular, relies on certain assumptions, such as linear light intensity, local smoothness and continuity [21], to function

effectively. However, when applied to high-dimensional, overparameterized feature embeddings produced by Vision Transformer blocks, it is unclear whether these assumptions still hold, or whether the operator remains effective.

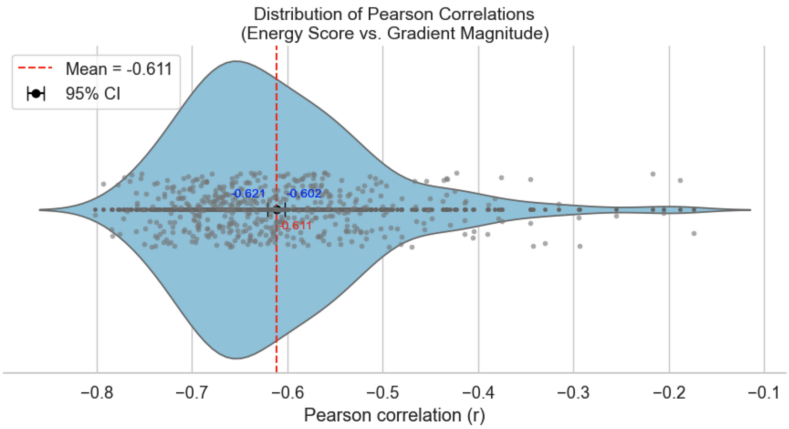


Figure 4.16: Correlation (Energy Score vs. Gradient Magnitude) with 95% confidence interval.

Our goal is to verify this empirically. The energy score used in *PiToMe* has demonstrated effectiveness across various backbones and tasks. Intuitively, tokens with higher energy scores tend to be less informative, typically corresponding to smooth or redundant regions such as background, while tokens with lower energy scores are often more salient or isolated, such as object boundaries. In these cases, we expect a negative correlation between energy scores and gradient magnitudes: low-energy tokens should exhibit high gradient magnitudes, and vice versa. To validate this, we computed the Pearson correlation coefficient between the energy score and the gradient magnitude for each image in the DIS5k dataset. We then plotted the distribution of these correlation values along with their 95% confidence interval. The results indicate a moderate negative correlation, with a mean coefficient of -0.61, providing statistical evidence that gradient magnitude is a reasonable proxy for the energy score. We also provide qualitative comparisons in **Figure 4.20**, where we visualize both energy scores and gradient magnitudes using heatmaps. These visualizations further support the observed negative correlation between the two measures in capturing token importance.

4.4.2 II: Revisiting Alternate Indexing for Token Partitioning

As discussed in **Section 3.3.2.2**, *PiToMe* employs alternate indexing over the token index array sorted by energy score. This strategy is based on the assumption that highly redundant tokens tend to cluster together in the sorted array. By alternately assigning these tokens to sets $\mathbb{A}$ and $\mathbb{B}$, *PiToMe* aims to evenly distribute similar tokens across the two sets, thereby increasing the likelihood of forming high-quality matched pairs. This pairing enhances the semantic preservation of the merged tokens and helps maintain segmentation performance. However, a key limitation of alternate indexing lies in its reduction ceiling. Under this scheme, the theoretical maximum token reduction rate is $r_{\max} = 50\%$, assuming that every token in set $\mathbb{A}$ is merged with its most similar counterpart in set $\mathbb{B}$. In our analysis, we observe that an aggressive merging rate does not necessarily degrade performance, as shown in **Figure 4.12**. In fact, forcing less similar tokens

being merged can sometimes lead to more enriched and generalized feature representations after unmerging, and potentially improve segmentation quality beyond what is achievable through the original model's learned representations. Thus, we aim to increase the reduction rate beyond 50% to further alleviate computational costs, while assessing its impact on segmentation performance.

To achieve this while still adhering to the core principle of bipartite soft matching (BSM), the size of the source set $\mathbb{A}$ should be increased and the destination set $\mathbb{B}$ correspondingly decreased. As discussed in **Section 4.3.2.3**, there is a trade-off in token partitioning: a smaller destination set improves computational efficiency, whereas a larger and representative destination set enhances merging effectiveness by improving match quality. Ideally, an effective token partitioning strategy should produce a reduced but semantically representative destination set $\mathbb{B}$, capable of preserving contextual richness and maintaining segmentation performance even under aggressive merging. To summarize, our goal is to construct a smaller yet contextually aware and representative destination set $\mathbb{B}$, and later the merging process can preserve semantically informative tokens and maintains spectral characteristics. This leads to the question of how energy scores or gradient magnitudes can effectively guide token partitioning to achieve these objectives.

A naive approach might be to select 25% of tokens at random or by uniformly sampling every fourth token from the energy or gradient-sorted index array. However, such methods do not guarantee spatial coverage or semantic representativeness across the entire image. To ensure that the large number of tokens in $\mathbb{A}$ have meaningful pairing opportunities, $\mathbb{B}$ should include representative tokens distributed across all image regions, particularly those corresponding to redundant patches. This necessitates a more structured sampling strategy. We adopt a grid-based token partitioning approach inspired by the method introduced in *ToMeSD*. Specifically, we incorporate stride-based spatial partitioning, in which the image grid is divided into fixed-size cells. In each cell, a single token is selected for inclusion in the destination set $\mathbb{B}$, and the remaining tokens are allocated to the source set $\mathbb{A}$. Unlike *ToMeSD*, where the destination token is either the upper-left corner or randomly selected within the cell, we utilize semantic information to guide this selection. The token with the highest energy score or the lowest gradient magnitude within each cell is chosen as the destination token, based on the intuition that it has the most redundant surrounding context. The remaining tokens in the cell are treated as source tokens to be merged.

Algorithm 4.1 Token Partitioning in *GradToMe* with Grid-Based Sampling

Require: Token tensor $T \in \mathbb{R}^{B\times H\times W\times C}$, Semantic map $S \in \mathbb{R}^{B\times H\times W\times 1}$, Strides s_x, s_y
Ensure: Src token set $\mathbb{A} \in \mathbb{R}^{B\times N_1\times C}$, Dst token set $\mathbb{B} \in \mathbb{R}^{B\times N_2\times C}$, where $N_1 + N_2 = H \cdot W$

1: Initialize empty sets $\mathbb{A}$ and $\mathbb{B}$
2: Partition the spatial grid of each image into non-overlapping cells of size $s_x \times s_y$, yielding $\frac{H}{s_y} \cdot \frac{W}{s_x}$ cells per image
3: In each cell, select the token corresponding to the **maximum** value in S if S represents energy scores, or the **minimum** value if S represents gradient magnitudes; assign these tokens to the destination set $\mathbb{B}$
4: Assign all remaining tokens within each cell to the source set $\mathbb{A}$
5: Return $\mathbb{A}, \mathbb{B}$

Using a default 2×2 stride configuration allows us to reach a theoretical maximum reduction rate of $r_{\max} = 75\%$. Even more aggressive rates are achievable by increasing the stride size, though prior findings from *ToMeSD* indicate that larger strides significantly degrade performance. We adopt the 2×2 configuration as a balanced choice, offering meaningful computational savings while maintaining quality. We evaluate this partitioning strategy using both energy score and gradient magnitude metrics and present the results in **Table 4.10**. Notably, when increasing the reduction rate from $r = 50\%$ to $r = 70\%$, average Mask IoU slightly improves, and overall segmentation quality preserves well. These findings confirm that our grid-based, context-aware partitioning strategy is both effective and practical for high-efficiency token merging in the SAM. We additionally conduct an ablation study on different sampling strategies, examining (1) the use of contextual information and (2) the application of grid-based sampling, to further validate our algorithmic design. The results, presented in **Table 4.11**, demonstrate the superiority of grid-based sampling combined with semantic awareness.

Model		**DIS**		**COIFT**		**HRSOD**		**ThinObject**		**Average**		(10^9)
		mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	mIoU↑	BIoU↑	**FLOPs/im**↓
SAM		0.5415	0.4611	0.8775	0.8204	0.8645	0.7816	0.5439	0.4605	0.7069	0.6309	486.42
SAM +GradToMe	r = 50%	0.5385 ↓0.0030	0.4570 ↓0.0041	0.8743 ↓0.0032	0.8189 ↓0.0015	0.8670 ↑0.0025	0.7855 ↑0.0039	0.5521 ↑0.0082	0.4651 ↑0.0046	0.7080 ↑0.0011	0.6316 ↑0.0007	424.09 ↓13%
	r = 60%	0.5410 ↓0.0005	0.4589 ↓0.0022	0.8718 ↓0.0057	0.8161 ↓0.0043	0.8692 ↑0.0047	0.7880 ↑0.0064	0.5495 ↑0.0056	0.4618 ↑0.0013	0.7079 ↑0.0010	0.6312 ↑0.0003	414.99 ↓15%
	r = 70%	0.5493 ↑0.0078	0.4639 ↑0.0028	0.8708 ↓0.0067	0.8129 ↓0.0075	0.8681 ↑0.0036	0.7862 ↑0.0046	0.5458 ↑0.0019	0.4578 ↓0.0027	0.7085 ↑0.0016	0.6302 ↓0.0007	406.84 ↓16%
SAM +PiToMe	r = 50%	0.5358 ↓0.0057	0.4542 ↓0.0069	0.8734 ↓0.0041	0.8183 ↓0.0021	0.8666 ↑0.0021	0.7850 ↑0.0034	0.5505 ↑0.0066	0.4639 ↑0.0034	0.7066 ↓0.0003	0.6304 ↓0.0005	425.96 ↓13%
	r = 60%	0.5411 ↓0.0004	0.4574 ↓0.0037	0.8727 ↓0.0048	0.8171 ↓0.0033	0.8673 ↑0.0028	0.7871 ↑0.0055	0.5483 ↑0.0044	0.4617 ↑0.0012	0.7074 ↑0.0005	0.6308 ↓0.0001	416.86 ↓14%
	r = 70%	0.5499 ↑0.0084	0.4646 ↑0.0035	0.8715 ↓0.0060	0.8138 ↓0.0066	0.8695 ↑0.0050	0.7881 ↑0.0065	0.5467 ↑0.0028	0.4646 ↑0.0041	0.7094 ↑0.0025	0.6328 ↑0.0019	408.71 ↓16%

Table 4.10: Performance comparison for *PiToMe* and *GradToMe* when increasing reduction rates with stride partition.

Contextual awareness	**Grid-based Sampling**	**Method**	**mIoU↑**	**bIoU↑**
✓	✓	PiToMe(Stride 2×2)	0.5499	0.4646
✓	✓	GradToMe(Stride 2×2)	0.5493 ↓-0.0006	0.4639 ↓-0.0007
✓	✗	GradToMe(Random 25%)	0.5478 ↓-0.0021	0.4635 ↓-0.0011
✓	✗	GradToMe(every 4th)	0.5466 ↓-0.0033	0.4618 ↓-0.0028
✗	✓	ToMeSD(Stride 2×2)	0.5441 ↓-0.0058	0.4594 ↓-0.0052
✗	✓	ToMeSD(Random 2×2)	0.5457 ↓-0.0042	0.4615 ↓-0.0031
✗	✗	ToMeSD(Random 25%)	0.5451 ↓-0.0048	0.4603 ↓-0.0043

Table 4.11: Ablation study of different sampling methods. Experiments are conducted with a reduction rate of $r = 70\%$ applied to the last five layers of SAM (ViT-B) on the DIS5K validation set.

Protection Mechanism As introduced in **Section 3.3.2.2**, *PiToMe* incorporates a special spectrum preservation mechanism, which protects tokens with the lowest energy scores by excluding them from the set of mergeable candidates. We consider this design choice potentially beneficial for preserving distinctive tokens and thereby maintaining final segmentation quality. To verify whether a similar protection mechanism is necessary, we conduct an ablation study. Specifically, we examine the effects of selecting or deselecting tokens with the lowest energy scores from the source token set $\mathbb{A}$ after token partitioning. In the deselection setting, these tokens are isolated and do not interact with the mergeable candidates. In the selection setting, they remain within the merge candidate set; if they exhibit sufficiently high similarity to tokens in the destination set $\mathbb{B}$, they are eligible to be merged during the greedy matching phase. As shown in **Table 4.12**, the configuration without protection, i.e., retaining tokens with the lowest energy scores in the source set $\mathbb{A}$, actually achieves better performance across varying reduction rates. In this setting, tokens in the source set can still be merged with their most similar counterparts, even if they have low energy scores and are normally considered important. This indicates that prioritizing the merging of highly similar tokens is more beneficial than preserving tokens solely based on their energy scores. Moreover, since destination tokens are selected as those with the highest energy scores within each region, a form of protection is inherently maintained: source tokens with higher energy scores are naturally more likely to have strong similarities with destination tokens. Therefore, we omit additional deselection-based protection mechanisms in our approach.

Reduction Rate	Protection	mIoU↑	BIoU↑
r = 50%	✓	0.5324	0.4508
	✗	**0.5358**	**0.4542**
r = 60%	✓	0.5339	0.4513
	✗	**0.5411**	**0.4574**
r = 70%	✓	0.5450	0.4600
	✗	**0.5499**	**0.4646**

Table 4.12: Ablation study of protection mechanisms. Experiments are conducted using *PiToMe* with stride partitioning applied to the last five layers of SAM (ViT-B), evaluated on the DIS5K validation set.

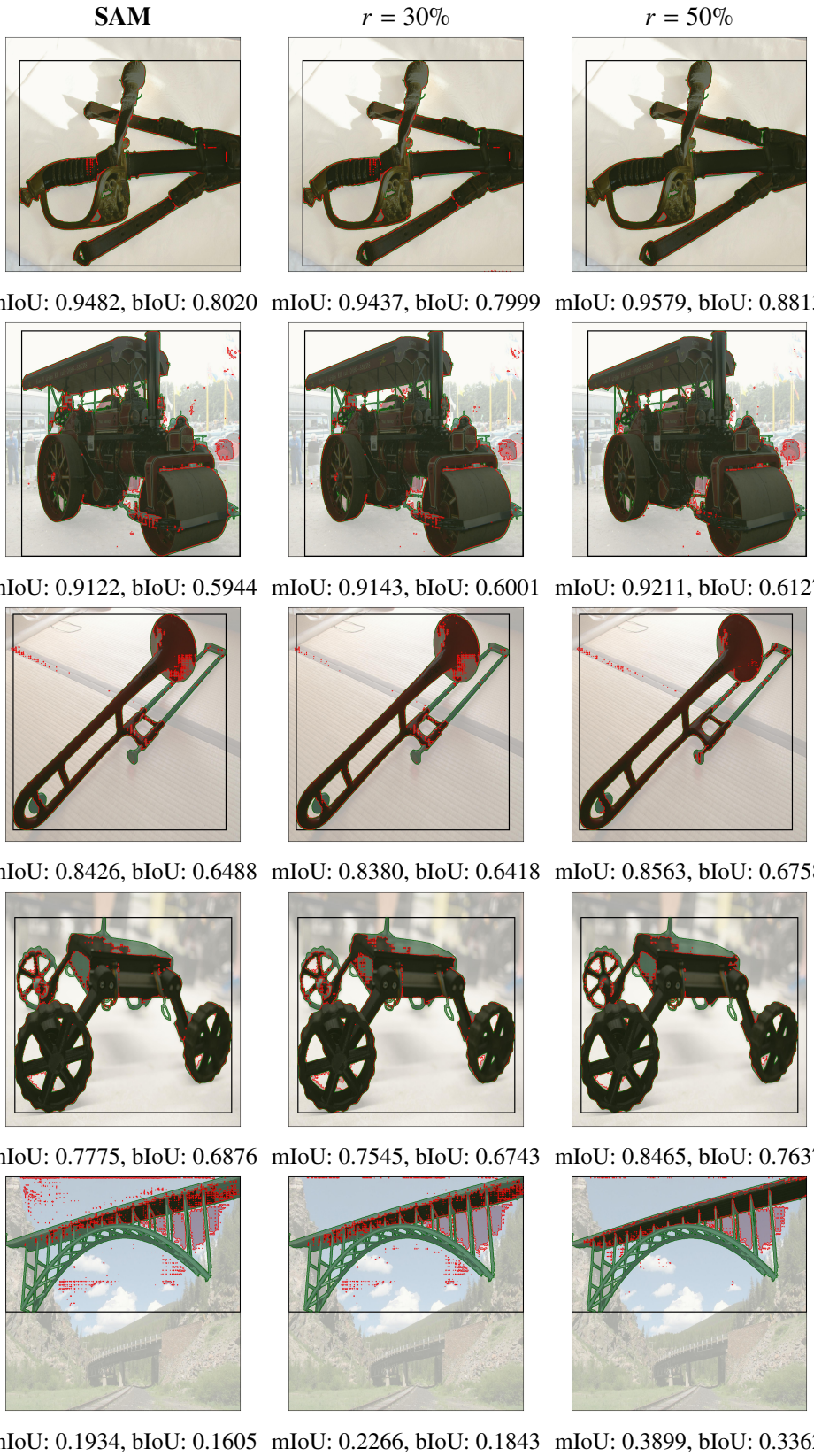


Figure 4.17: Qualitative comparisons of different reduction rates r when applying *PiToMe* in last 5 layers.

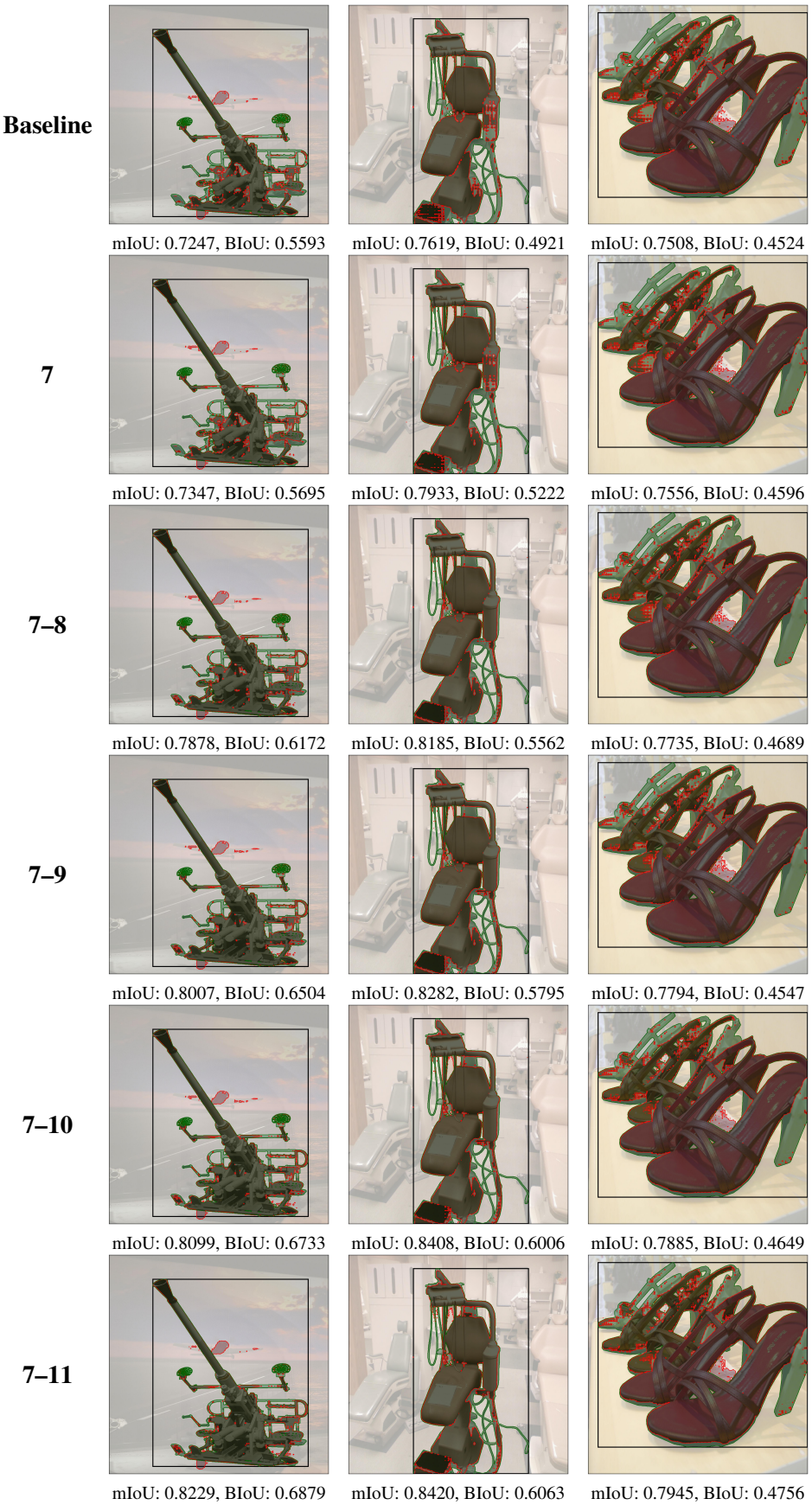


Figure 4.18: Qualitative comparisons of successive token merging from 7th layer to 11th layer using *PiToMe*.

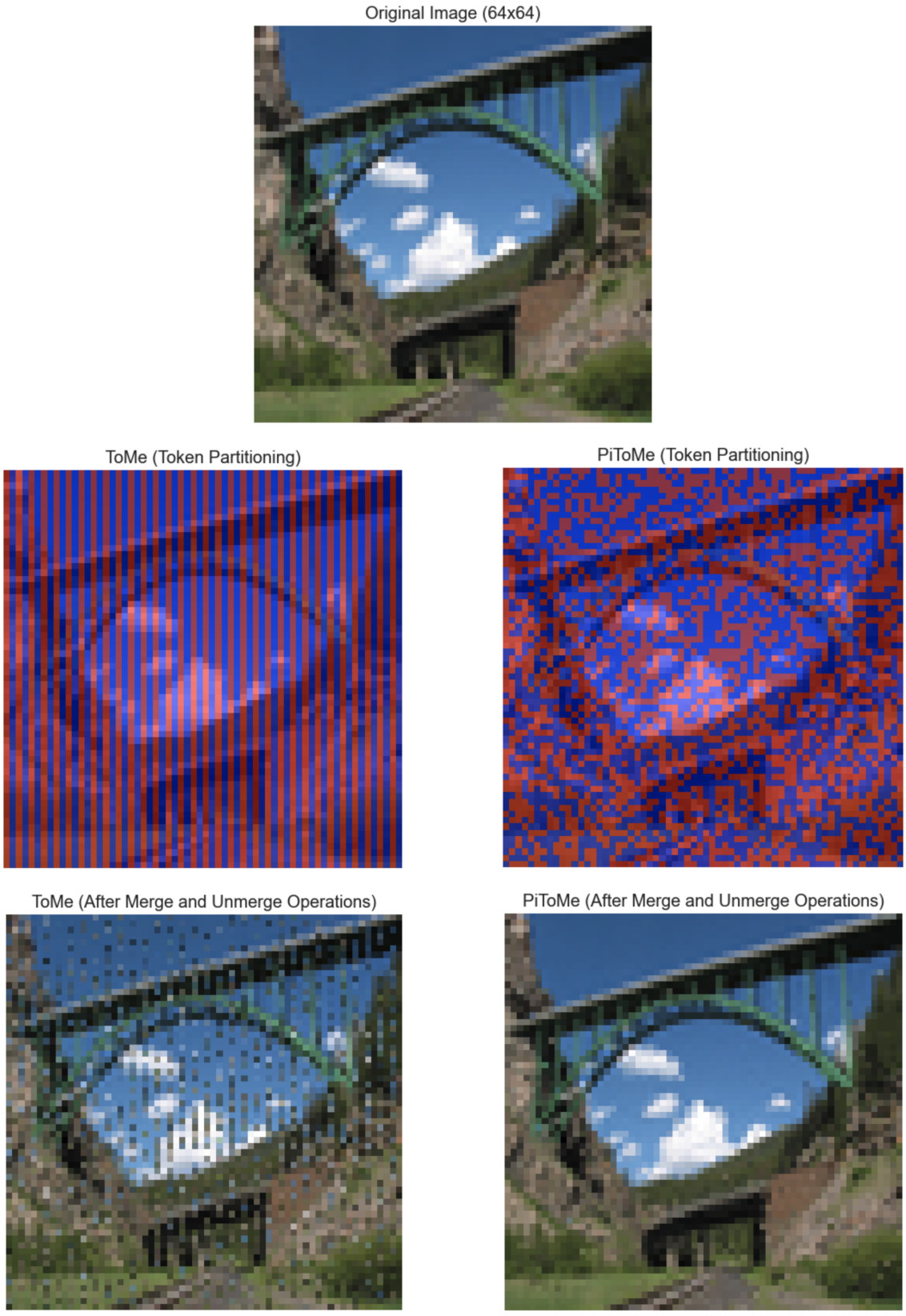


Figure 4.19: A qualitative comparison between token partitioning by *ToMe* and *PiToMe*.

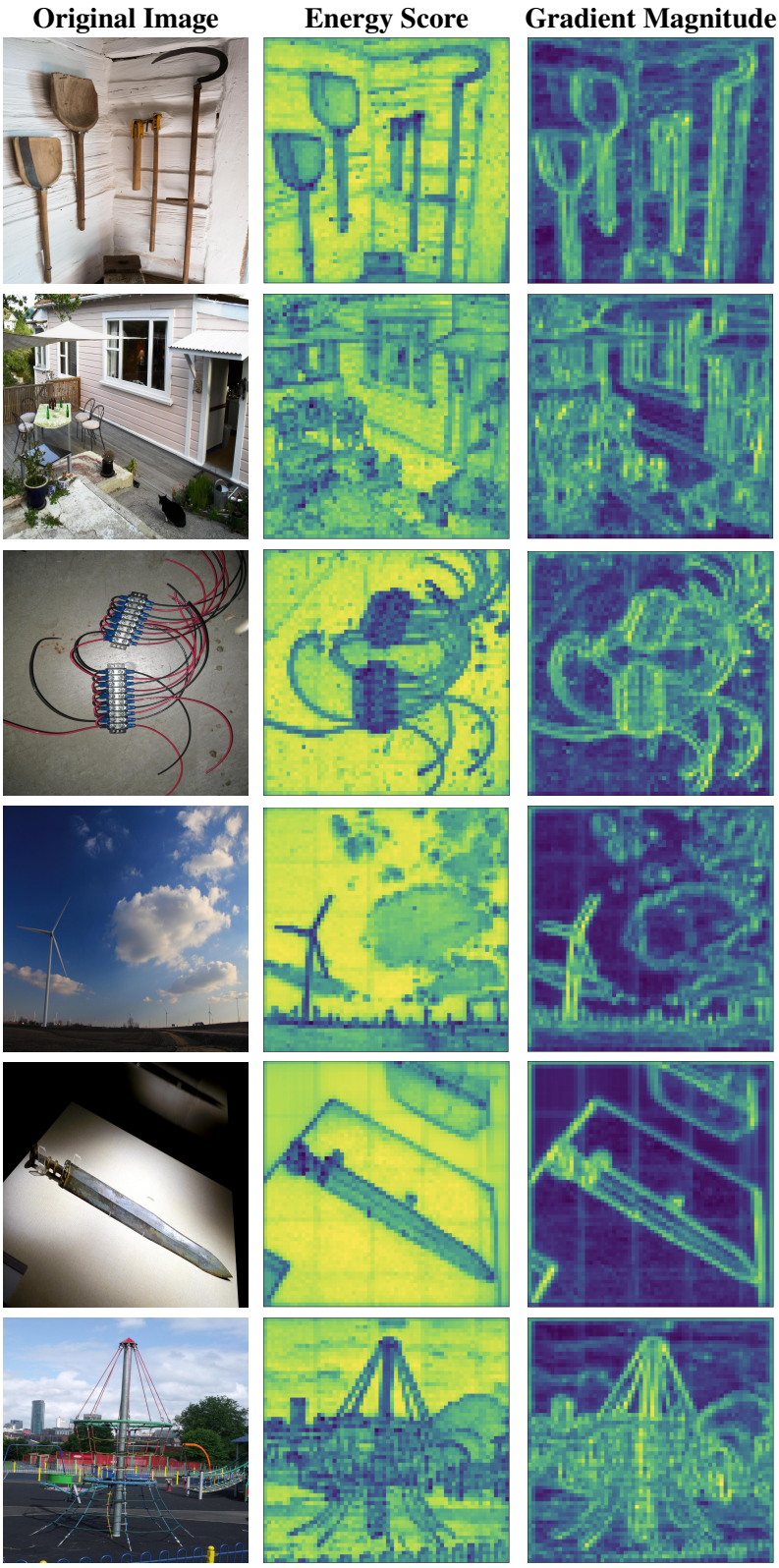


Figure 4.20: Qualitative comparisons of original images, and heatmap of energy scores and gradient magnitudes by Sobel operator.

5 Discussion and Future Work

Vision Transformer (ViT)-based models such as the Segment Anything Model (SAM) have demonstrated exceptional capabilities in segmentation tasks, largely due to their ability to capture long-range dependencies and rich semantic features. However, these strengths come at a cost: ViT architectures are resource-intensive, posing significant challenges for deployment in latency-sensitive or resource-constrained environments.

This thesis was motivated by the need to alleviate the computational overhead of SAM by exploring and extending token merging strategies—training-free techniques that reduce the number of tokens by merging similar ones, thereby minimizing redundant computations while maintaining model performance. We began with a comprehensive analysis of SAM's architecture, with particular focus on its image encoder. Unlike standard ViTs, SAM features a more complex and specialized design that, while contributing to its outstanding performance, introduces unique challenges for token reduction. To address these, we adapted several BSM-based token merging techniques to suit SAM's encoder and investigated a wide range of merging configurations. Our empirical evaluation across extremely fine-grained and complex datasets reveals that token merging not only improves computational efficiency but, in certain configurations, can also enhance segmentation accuracy. We conducted in-depth analysis to understand the underlying reasons behind this unexpected performance gain, supported by qualitative and quantitative evidence. Furthermore, our findings suggest that contextual awareness plays a key role in guiding token partition and thus enhances the effectiveness of token merging.

Building on these insights, we introduced *GradToMe*, a token merging algorithm developed to address the limitations of *PiToMe* and broaden the solution space available for token reduction. *GradToMe* incorporates two key innovations: (1) a computationally efficient gradient approximation method that replaces the energy score for ranking token importance, and (2) a grid-based sampling strategy that replaces alternating indexing to construct a compact yet representative destination set, enabling more aggressive token reduction. Experimental results confirm the effectiveness and greater efficiency gains of applying *GradToMe* within the SAM's image encoder.

Outlook

While our contributions provide a solid foundation, several promising directions remain for future exploration:

- **Reduction Rate r Scheduling as a Hyperparameter:** In our experimental setup, aggressive reduction rates were manually selected to maximize efficiency gains. However, as previously discussed, the reduction rate r, which governs the extent of token merging, significantly impacts model performance and should ideally vary across different layers. Moreover, *GradToMe* demonstrates potential as a candidate for more aggressive merging while preserving

segmentation quality. Thus, r can be considered as a hyperparameter of the model architecture merely a parameter associated with token merging algorithms, similar to the drop rate in dense neural networks. Future work could investigate systematic and structural approaches for reduction rate scheduling throughout the model depth, aiming to achieve an optimal trade-off between computational efficiency and segmentation accuracy.

- **Generalization Across ViT Architectures:** Our current findings on the beneficial effects of token merging, such as improved segmentation accuracy, have been validated using SAM's image encoder. To ensure these effects are not solely due to SAM's specialized architecture, further investigations are needed on standard ViT models (e.g., Segmenter [71], SETR [100]) to confirm the broader applicability of our hypotheses.
- **Cross-Task Generalization and Zero-Shot Evaluation:** SAM is known for its powerful zero-shot generalization across various segmentation tasks. Our experiments focused on promptable segmentation with regular bounding box prompts. Future work should explore whether SAM integrated with token merging retains its strong zero-shot transfer capabilities across diverse tasks such as instance segmentation, interactive segmentation, or panoptic segmentation.
- **Prompt Robustness:** A distinctive strength of SAM is its ability to produce meaningful masks even from ambiguous prompts. However, our evaluation was limited to structured box prompts. Additional research should explore how token merging affects prompt robustness and whether the resulting image embeddings continue to produce semantically rich and consistent segmentation masks in more open-ended or user-driven settings.

Through these directions, we hope to further unlock the potential of token merging as a practical and principled approach to vision transformer models, bridging the gap between high-performing foundation models and real-time usability through targeted efficiency improvements.

Bibliography

[1] A. S. Asratian, T. M. J. Denley, R. Häggkvist. *Bipartite Graphs and Their Applications*. Vol. 131. Cambridge Tracts in Mathematics. Cambridge: Cambridge University Press, 1998 (cit. on p. 26).

[2] R. Balakrishnan. "The energy of a graph". In: *Linear Algebra and Its Applications - LINEAR ALGEBRA APPL* 387 (Aug. 2004), pp. 287–295. DOI: 10.1016/j.laa.2004.02.038 (cit. on pp. 28, 29).

[3] D. Bolya, C. Fu, X. Dai, P. Zhang, C. Feichtenhofer, J. Hoffman. "Token Merging: Your ViT But Faster". In: *CoRR* abs/2210.09461 (2022). DOI: 10.48550/ARXIV.2210.09461. arXiv: 2210.09461. URL: https://doi.org/10.48550/arXiv.2210.09461 (cit. on pp. 13, 14, 16, 25, 27, 41, 44, 48, 49, 51, 57).

[4] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, J. Hoffman. *Hydra Attention: Efficient Attention with Many Heads*. 2022. arXiv: 2209.07484 [cs.CV]. URL: https://arxiv.org/abs/2209.07484 (cit. on p. 15).

[5] D. Bolya, J. Hoffman. "Token Merging for Fast Stable Diffusion". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023, pp. 4599–4603. DOI: 10.1109/CVPRW59228.2023.00484 (cit. on pp. 13, 14, 16, 17, 27–29, 41, 44, 48, 57).

[6] M. Bonnaerens, J. Dambre. *Learned Thresholds Token Merging and Pruning for Vision Transformers*. 2023. arXiv: 2307.10780 [cs.CV]. URL: https://arxiv.org/abs/2307.10780 (cit. on pp. 13, 16).

[7] Y. Boykov, O. Veksler, R. Zabih. "Fast approximate energy minimization via graph cuts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (2001), pp. 1222–1239. DOI: 10.1109/34.969114 (cit. on p. 25).

[8] Q. Cao, B. Paranjape, H. Hajishirzi. *PuMer: Pruning and Merging Tokens for Efficient Vision Language Models*. 2023. arXiv: 2305.17530 [cs.CV]. URL: https://arxiv.org/abs/2305.17530 (cit. on pp. 13, 16).

[9] S. Chang, P. Wang, M. Lin, F. Wang, D. J. Zhang, R. Jin, M. Z. Shou. *Making Vision Transformers Efficient from A Token Sparsification View*. 2023. arXiv: 2303.08685 [cs.CV]. URL: https://arxiv.org/abs/2303.08685 (cit. on pp. 13, 16).

[10] C.-F. Chen, Q. Fan, R. Panda. *CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification*. 2021. arXiv: 2103.14899 [cs.CV]. URL: https://arxiv.org/abs/2103.14899 (cit. on p. 13).

[11] M. Chen, W. Shao, P. Xu, M. Lin, K. Zhang, F. Chao, R. Ji, Y. Qiao, P. Luo. *DiffRate : Differentiable Compression Rate for Efficient Vision Transformers*. 2023. arXiv: 2305.17997 [cs.CV]. URL: https://arxiv.org/abs/2305.17997 (cit. on pp. 13, 16, 48).

[12] Q. Chen, Q. Wu, J. Wang, Q. Hu, T. Hu, E. Ding, J. Cheng, J. Wang. *MixFormer: Mixing Features across Windows and Dimensions*. Apr. 2022. DOI: 10.48550/arXiv.2204.02557 (cit. on p. 38).

[13] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, Y. Qiao. *Vision Transformer Adapter for Dense Predictions*. 2023. arXiv: 2205.08534 [cs.CV]. URL: https://arxiv.org/abs/2205.08534 (cit. on p. 42).

[14] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, R. Girdhar. *Masked-attention Mask Transformer for Universal Image Segmentation*. 2022. arXiv: 2112.01527 [cs.CV]. URL: https://arxiv.org/abs/2112.01527 (cit. on p. 13).

[15] D.-A. Clevert, T. Unterthiner, S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *ICLR (Poster)*. Ed. by Y. Bengio, Y. LeCun. 2016. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#ClevertUH15 (cit. on p. 29).

[16] J. Devlin, M. Chang, K. Lee, K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (cit. on p. 19).

[17] Y. Ding, H. Qin, Q. Yan, Z. Chai, J. Liu, X. Wei, X. Liu. "Towards Accurate Post-Training Quantization for Vision Transformer". In: *Proceedings of the 30th ACM International Conference on Multimedia*. MM '22. ACM, Oct. 2022, pp. 5380–5388. DOI: 10.1145/3503161.3547826. URL: http://dx.doi.org/10.1145/3503161.3547826 (cit. on p. 15).

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929 (cit. on pp. 13, 19).

[19] P. Esser, R. Rombach, B. Ommer. *Taming Transformers for High-Resolution Image Synthesis*. 2021. arXiv: 2012.09841 [cs.CV]. URL: https://arxiv.org/abs/2012.09841 (cit. on p. 13).

[20] M. Fayyaz, S. A. Koohpayegani, F. R. Jafari, S. Sengupta, H. R. V. Joze, E. Sommerlade, H. Pirsiavash, J. Gall. *Adaptive Token Sampling For Efficient Vision Transformers*. 2022. arXiv: 2111.15667 [cs.CV]. URL: https://arxiv.org/abs/2111.15667 (cit. on pp. 13, 16).

[21] R. C. Gonzalez, R. E. Woods. *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X. URL: http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X (cit. on pp. 32, 63).

[22] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, R. B. Girshick. "Masked Autoencoders Are Scalable Vision Learners". In: *CoRR* abs/2111.06377 (2021). arXiv: 2111.06377. URL: https://arxiv.org/abs/2111.06377 (cit. on p. 24).

[23] K. He, X. Zhang, S. Ren, J. Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (cit. on pp. 19, 36).

[24] J. H. Heo, A. Fayyazi, M. Nazemi, M. Pedram. "A Fast Training-Free Compression Framework for Vision Transformers". In: *arXiv preprint arXiv:2303.02331* (2023) (cit. on pp. 13, 16).

[25] G. Hinton, O. Vinyals, J. Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML]. URL: https://arxiv.org/abs/1503.02531 (cit. on p. 15).

[26] S. Hochreiter, J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 19).

[27] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. 2017. arXiv: 1712.05877 [cs.LG]. URL: https://arxiv.org/abs/1712.05877 (cit. on p. 15).

[28] P. Jin, B. Zhu, L. Yuan, S. Yan. *MoH: Multi-Head Attention as Mixture-of-Head Attention*. 2024. arXiv: 2410.11842 [cs.CV]. URL: https://arxiv.org/abs/2410.11842 (cit. on p. 50).

[29] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, F. Yu. “Segment anything in high quality”. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS ’23. New Orleans, LA, USA: Curran Associates Inc., 2023 (cit. on p. 43).

[30] D. Kienzle, M. Kantonis, R. Schön, R. Lienhart. *Segformer++: Efficient Token-Merging Strategies for High-Resolution Semantic Segmentation*. 2024. arXiv: 2405.14467 [cs.CV]. URL: https://arxiv.org/abs/2405.14467 (cit. on pp. 13, 16, 17, 31, 41, 44, 48, 49, 57).

[31] M. Kim, S. Gao, Y.-C. Hsu, Y. Shen, H. Jin. *Token Fusion: Bridging the Gap between Token Pruning and Token Merging*. 2023. arXiv: 2312.01026 [cs.CV]. URL: https://arxiv.org/abs/2312.01026 (cit. on pp. 13, 16, 48).

[32] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, R. Girshick. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: https://arxiv.org/abs/2304.02643 (cit. on pp. 14, 23, 24, 36, 44).

[33] Z. Kong, P. Dong, X. Ma, X. Meng, M. Sun, W. Niu, X. Shen, G. Yuan, B. Ren, M. Qin, H. Tang, Y. Wang. *SPViT: Enabling Faster Vision Transformers via Soft Token Pruning*. 2022. arXiv: 2112.13890 [cs.CV]. URL: https://arxiv.org/abs/2112.13890 (cit. on pp. 13, 16).

[34] J. Koo, J. Yang, L. An, G. Sergio, S. I. Park. *Swin-Free: Achieving Better Cross-Window Attention and Efficiency with Size-varying Window*. June 2023. DOI: 10.48550/arXiv.2306.13776 (cit. on p. 38).

[35] S. A. Koohpayegani, H. Pirsiavash. *SimA: Simple Softmax-free Attention for Vision Transformers*. 2024. arXiv: 2206.08898 [cs.CV]. URL: https://arxiv.org/abs/2206.08898 (cit. on p. 15).

[36] Y. Lecun, Y. Bengio. “Convolutional Networks for Images, Speech, and Time-Series”. In: Jan. 1995 (cit. on p. 19).

[37] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007. DOI: 10.1137/1.9780898717839. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717839. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898717839 (cit. on p. 32).

[38] J. Li, R. Cotterell, M. Sachan. *Differentiable Subset Pruning of Transformer Heads*. 2023. arXiv: 2108.04657 [cs.CL]. URL: https://arxiv.org/abs/2108.04657 (cit. on p. 50).

[39] J. Li, Y. Wang, X. Zhang, B. Shi, D. Jiang, C. Li, W. Dai, H. Xiong, Q. Tian. *AiluRus: A Scalable ViT Framework for Dense Prediction*. 2023. arXiv: 2311.01197 [cs.CV]. URL: https://arxiv.org/abs/2311.01197 (cit. on pp. 13, 16).

[40] Y. Li, C. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, C. Feichtenhofer. "Improved Multiscale Vision Transformers for Classification and Detection". In: *CoRR* abs/2112.01526 (2021). arXiv: 2112.01526. URL: https://arxiv.org/abs/2112.01526 (cit. on pp. 13, 40).

[41] Z. Li, M. Sun, A. Lu, H. Ma, G. Yuan, Y. Xie, H. Tang, Y. Li, M. Leeser, Z. Wang, X. Lin, Z. Fang. *Auto-ViT-Acc: An FPGA-Aware Automatic Acceleration Framework for Vision Transformer with Mixed-Scheme Quantization*. 2022. arXiv: 2208.05163 [cs.CV]. URL: https://arxiv.org/abs/2208.05163 (cit. on p. 15).

[42] W. Liang, Y. Yuan, H. Ding, X. Luo, W. Lin, D. Jia, Z. Zhang, C. Zhang, H. Hu. *Expediting Large-Scale Vision Transformer for Dense Prediction without Fine-tuning*. 2022. arXiv: 2210.01035 [cs.CV]. URL: https://arxiv.org/abs/2210.01035 (cit. on pp. 13, 16).

[43] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, P. Xie. *Not All Patches are What You Need: Expediting Vision Transformers via Token Reorganizations*. 2022. arXiv: 2202.07800 [cs.CV]. URL: https://arxiv.org/abs/2202.07800 (cit. on pp. 13, 16).

[44] J. H. Liew, S. Cohen, B. Price, L. Mai, J. Feng. "Deep Interactive Thin Object Selection". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 305–314. DOI: 10.1109/WACV48630.2021.00035 (cit. on p. 43).

[45] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, S. J. Belongie. "Feature Pyramid Networks for Object Detection". In: *CoRR* abs/1612.03144 (2016). arXiv: 1612.03144. URL: http://arxiv.org/abs/1612.03144 (cit. on p. 36).

[46] X. Liu, H. Peng, N. Zheng, Y. Yang, H. Hu, Y. Yuan. *EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention*. 2023. arXiv: 2305.07027 [cs.CV]. URL: https://arxiv.org/abs/2305.07027 (cit. on p. 15).

[47] Y. Liu, H. Yang, Z. Dong, K. Keutzer, L. Du, S. Zhang. *NoisyQuant: Noisy Bias-Enhanced Post-Training Activation Quantization for Vision Transformers*. 2023. arXiv: 2211.16056 [cs.CV]. URL: https://arxiv.org/abs/2211.16056 (cit. on p. 15).

[48] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *CoRR* abs/2103.14030 (2021). arXiv: 2103.14030. URL: https://arxiv.org/abs/2103.14030 (cit. on pp. 13, 15, 38).

[49] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 25).

[50] S. Long, Z. Zhao, J. Pi, S. Wang, J. Wang. *Beyond Attentive Tokens: Incorporating Token Importance and Diversity for Efficient Vision Transformers*. 2022. arXiv: 2211.11315 [cs.CV]. URL: https://arxiv.org/abs/2211.11315 (cit. on pp. 13, 16).

[51] J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, L. Zhang. *SOFT: Softmax-free Transformer with Linear Complexity*. 2022. arXiv: 2110.11945 [cs.CV]. URL: https://arxiv.org/abs/2110.11945 (cit. on p. 15).

[52] D. Marin, J. R. Chang, A. Ranjan, A. Prabhu, M. Rastegari, O. Tuzel. "Token Pooling in Vision Transformers". In: *CoRR* abs/2110.03860 (2021). arXiv: 2110.03860. URL: https://arxiv.org/abs/2110.03860 (cit. on pp. 13, 16).

[53] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, S.-N. Lim. *AdaViT: Adaptive Vision Transformers for Efficient Image Recognition*. 2021. arXiv: 2111.15668 [cs.CV]. URL: https://arxiv.org/abs/2111.15668 (cit. on pp. 13, 16).

[54] N. Norouzi, S. Orlova, D. de Geus, G. Dubbelman. *ALGM: Adaptive Local-then-Global Token Merging for Efficient Semantic Segmentation with Plain Vision Transformers*. 2024. arXiv: 2406.09936 [cs.CV]. URL: https://arxiv.org/abs/2406.09936 (cit. on pp. 13, 16, 17, 41, 46, 48, 49, 59).

[55] Z. Pan, B. Zhuang, H. He, J. Liu, J. Cai. *Less is More: Pay Less Attention in Vision Transformers*. 2021. arXiv: 2105.14217 [cs.CV]. URL: https://arxiv.org/abs/2105.14217 (cit. on pp. 13, 16).

[56] W. Peebles, S. Xie. *Scalable Diffusion Models with Transformers*. 2023. arXiv: 2212.09748 [cs.CV]. URL: https://arxiv.org/abs/2212.09748 (cit. on p. 13).

[57] PyTorch Team. *FLOPs Estimation with fvcore*. https://detectron2.readthedocs.io/en/latest/modules/fvcore.html#fvcore.nn.FlopCountAnalysis. Accessed: 2025-04-09 (cit. on p. 43).

[58] X. Qin, H. Dai, X. Hu, D.-P. Fan, L. Shao, L. V. Gool. *Highly Accurate Dichotomous Image Segmentation*. 2022. arXiv: 2203.03041 [cs.CV]. URL: https://arxiv.org/abs/2203.03041 (cit. on p. 43).

[59] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever. "Improving language understanding by generative pre-training". In: (2018) (cit. on p. 19).

[60] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, C.-J. Hsieh. *DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification*. 2021. arXiv: 2106.02034 [cs.CV]. URL: https://arxiv.org/abs/2106.02034 (cit. on pp. 13, 16).

[61] S. Ren, Z. Gao, T. Hua, Z. Xue, Y. Tian, S. He, H. Zhao. *Co-advise: Cross Inductive Bias Distillation*. 2021. arXiv: 2106.12378 [cs.CV]. URL: https://arxiv.org/abs/2106.12378 (cit. on p. 15).

[62] C. Renggli, A. S. Pinto, N. Houlsby, B. Mustafa, J. Puigcerver, C. Riquelme. *Learning to Merge Tokens in Vision Transformers*. 2022. arXiv: 2202.12015 [cs.CV]. URL: https://arxiv.org/abs/2202.12015 (cit. on pp. 13, 16).

[63] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. URL: https://github.com/CompVis/latent-diffusionhttps://arxiv.org/abs/2112.10752 (cit. on pp. 27, 31).

[64] O. Ronneberger, P. Fischer, T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (cit. on p. 31).

[65] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, A. Angelova. "TokenLearner: adaptive space-time tokenization for videos". In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021. ISBN: 9781713845393 (cit. on pp. 13, 16).

[66] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, A. Angelova. *TokenLearner: What Can 8 Learned Tokens Do for Images and Videos?* 2022. arXiv: 2106.11297 [cs.CV]. URL: https://arxiv.org/abs/2106.11297 (cit. on pp. 13, 16).

[67] P. Shaw, J. Uszkoreit, A. Vaswani. "Self-Attention with Relative Position Representations". In: *CoRR* abs/1803.02155 (2018). arXiv: 1803.02155. URL: http://arxiv.org/abs/1803.02155 (cit. on p. 40).

[68] D. Shi, C. Tao, A. Rao, Z. Yang, C. Yuan, J. Wang. *CrossGET: Cross-Guided Ensemble of Tokens for Accelerating Vision-Language Transformers*. 2024. arXiv: 2305.17455 [cs.CV]. URL: https://arxiv.org/abs/2305.17455 (cit. on pp. 13, 16, 48).

[69] S. W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Available at www.dspguide.com. California Technical Publishing, 1997. URL: http://www.dspguide.com (cit. on p. 32).

[70] Z. Song, J. Yu, Y.-P. P. Chen, W. Yang. *Transformer Tracking with Cyclic Shifting Window Attention*. May 2022. DOI: 10.48550/arXiv.2205.03806 (cit. on p. 38).

[71] R. Strudel, R. Garcia, I. Laptev, C. Schmid. "Segmenter: Transformer for Semantic Segmentation". In: *CoRR* abs/2105.05633 (2021). arXiv: 2105.05633. URL: https://arxiv.org/abs/2105.05633 (cit. on pp. 13, 17, 48, 74).

[72] Q. Tang, B. Zhang, J. Liu, F. Liu, Y. Liu. *Dynamic Token Pruning in Plain Vision Transformers for Semantic Segmentation*. 2023. arXiv: 2308.01045 [cs.CV]. URL: https://arxiv.org/abs/2308.01045 (cit. on pp. 13, 16).

[73] Q. Tang, B. Zhang, J. Liu, F. Liu, Y. Liu. *Dynamic Token Pruning in Plain Vision Transformers for Semantic Segmentation*. 2023. arXiv: 2308.01045 [cs.CV]. URL: https://arxiv.org/abs/2308.01045 (cit. on p. 16).

[74] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou. *Training data-efficient image transformers distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV]. URL: https://arxiv.org/abs/2012.12877 (cit. on pp. 13, 15).

[75] H.-C. Tran, D. M. H. Nguyen, D. M. Nguyen, T. Nguyen, N. Le, P. Xie, D. Sonntag, J. Y. Zou, B. T. Nguyen, M. Niepert. "Accelerating Transformers with Spectrum-Preserving Token Merging." In: *CoRR* abs/2405.16148 (2024). URL: http://dblp.uni-trier.de/db/journals/corr/corr2405.html#abs-2405-16148 (cit. on pp. 13, 14, 16, 27, 41, 44, 48, 57).

[76] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 13, 19, 21).

[77] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, I. Titov. *Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned*. 2019. arXiv: 1905.09418 [cs.CL]. URL: https://arxiv.org/abs/1905.09418 (cit. on p. 50).

[78] H. Wang, B. Dedhia, N. K. Jha. *Zero-TPrune: Zero-Shot Token Pruning through Leveraging of the Attention Graph in Pre-Trained Transformers*. 2024. arXiv: 2305.17328 [cs.CV]. URL: https://arxiv.org/abs/2305.17328 (cit. on pp. 13, 16).

[79] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao. *Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions*. 2021. arXiv: 2102.12122 [cs.CV]. URL: https://arxiv.org/abs/2102.12122 (cit. on pp. 13, 15).

[80] K. Wu, J. Zhang, H. Peng, M. Liu, B. Xiao, J. Fu, L. Yuan. *TinyViT: Fast Pretraining Distillation for Small Vision Transformers*. 2022. arXiv: 2207.10666 [cs.CV]. URL: https://arxiv.org/abs/2207.10666 (cit. on p. 15).

[81] X. Wu, F. Zeng, X. Wang, X. Chen. *PPT: Token Pruning and Pooling for Efficient Vision Transformers*. 2024. arXiv: 2310.01812 [cs.CV]. URL: https://arxiv.org/abs/2310.01812 (cit. on pp. 13, 16).

[82] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Álvarez, P. Luo. "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers". In: *CoRR* abs/2105.15203 (2021). arXiv: 2105.15203. URL: https://arxiv.org/abs/2105.15203 (cit. on pp. 13, 17, 31, 42).

[83] Y. Xiong, B. Varadarajan, L. Wu, X. Xiang, F. Xiao, C. Zhu, X. Dai, D. Wang, F. Sun, F. Iandola, R. Krishnamoorthi. "EfficientSAM: Leveraged Masked Image Pretraining for Efficient Segment Anything". In: June 2024, pp. 16111–16121. DOI: 10.1109/CVPR52733.2024.01525 (cit. on p. 36).

[84] S. Xu, H. Yuan, Q. Shi, L. Qi, J. Wang, Y. Yang, Y. Li, K. Chen, Y. Tong, B. Ghanem, X. Li, M.-H. Yang. "RMP-SAM: Towards Real-Time Multi-Purpose Segment Anything". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=1pXzC30ry5 (cit. on p. 36).

[85] H. Yan, M. Wu, C. Zhang. "Multi-Scale Representations by Varying Window Attention for Semantic Segmentation". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=lAhWGOkpSR (cit. on p. 38).

[86] H. Yang, H. Yin, M. Shen, P. Molchanov, H. Li, J. Kautz. *Global Vision Transformer Pruning with Hessian-Aware Saliency*. 2023. arXiv: 2110.04869 [cs.CV]. URL: https://arxiv.org/abs/2110.04869 (cit. on p. 15).

[87] H. Yin, A. Vahdat, J. Alvarez, A. Mallya, J. Kautz, P. Molchanov. *AdaViT: Adaptive Tokens for Efficient Vision Transformer*. 2022. arXiv: 2112.07658 [cs.CV]. URL: https://arxiv.org/abs/2112.07658 (cit. on p. 16).

[88] H. You, Y. Xiong, X. Dai, B. Wu, P. Zhang, H. Fan, P. Vajda, Y. C. Lin. *Castling-ViT: Compressing Self-Attention via Switching Towards Linear-Angular Attention at Vision Transformer Inference*. 2024. arXiv: 2211.10526 [cs.CV]. URL: https://arxiv.org/abs/2211.10526 (cit. on p. 15).

[89] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, L. Cui. "Width & Depth Pruning for Vision Transformers". In: *AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 2022. 2022 (cit. on p. 15).

[90] L. Yu, W. Xiang. *X-Pruner: eXplainable Pruning for Vision Transformers*. 2023. arXiv: 2303.04935 [cs.CV]. URL: https://arxiv.org/abs/2303.04935 (cit. on p. 15).

[91] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, S. Yan. *MetaFormer Is Actually What You Need for Vision*. 2022. arXiv: 2111.11418 [cs.CV]. URL: https://arxiv.org/abs/2111.11418 (cit. on p. 15).

[92] Z. Yuan, C. Xue, Y. Chen, Q. Wu, G. Sun. *PTQ4ViT: Post-training quantization for vision transformers with twin uniform quantization*. 2024. arXiv: 2111.12293 [cs.CV]. URL: https://arxiv.org/abs/2111.12293 (cit. on p. 15).

[93] Y. Zeng, P. Zhang, J. Zhang, Z. Lin, H. Lu. *Towards High-Resolution Salient Object Detection*. 2019. arXiv: 1908.07274 [cs.CV]. URL: https://arxiv.org/abs/1908.07274 (cit. on p. 43).

[94] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, B. Guo. *StyleSwin: Transformer-based GAN for High-resolution Image Generation*. 2022. arXiv: 2112.10762 [cs.CV]. URL: https://arxiv.org/abs/2112.10762 (cit. on p. 13).

[95] C. Zhang, D. Han, Y. Qiao, J. Kim, S.-H. Bae, S. Lee, C. S. Hong. *Faster Segment Anything: Towards Lightweight SAM for Mobile Applications*. June 2023. DOI: 10.48550/arXiv.2306.14289 (cit. on p. 36).

[96] J. Zhang, H. Peng, K. Wu, M. Liu, B. Xiao, J. Fu, L. Yuan. *MiniViT: Compressing Vision Transformers with Weight Multiplexing*. 2022. arXiv: 2204.07154 [cs.CV]. URL: https://arxiv.org/abs/2204.07154 (cit. on p. 15).

[97] Q. Zhang, Y. Xu, J. Zhang, D. Tao. *ViTAEv2: Vision Transformer Advanced by Exploring Inductive Bias for Image Recognition and Beyond*. 2022. arXiv: 2202.10108 [cs.CV]. URL: https://arxiv.org/abs/2202.10108 (cit. on p. 13).

[98] Q. Zhang, Y. Xu, J. Zhang, D. Tao. "VSA: Learning Varied-Size Window Attention in Vision Transformers". In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV*. Tel Aviv, Israel: Springer-Verlag, 2022, pp. 466–483. ISBN: 978-3-031-19805-2. DOI: 10.1007/978-3-031-19806-9_27. URL: https://doi.org/10.1007/978-3-031-19806-9_27 (cit. on p. 38).

[99] W. Zhang, Z. Huang, G. Luo, T. Chen, X. Wang, W. Liu, G. Yu, C. Shen. "TopFormer: Token Pyramid Transformer for Mobile Semantic Segmentation". In: (2022) (cit. on p. 36).

[100] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. S. Torr, L. Zhang. "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers". In: *CoRR* abs/2012.15840 (2020). arXiv: 2012.15840. URL: https://arxiv.org/abs/2012.15840 (cit. on pp. 13, 17, 74).

[101] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba. "Semantic Understanding of Scenes through the ADE20K Dataset". In: *CoRR* abs/1608.05442 (2016). arXiv: 1608.05442. URL: http://arxiv.org/abs/1608.05442 (cit. on p. 48).

[102] Z. Zong, K. Li, G. Song, Y. Wang, Y. Qiao, B. Leng, Y. Liu. *Self-slimmed Vision Transformer*. 2022. arXiv: 2111.12624 [cs.CV]. URL: https://arxiv.org/abs/2111.12624 (cit. on pp. 13, 16).

All links were last followed on April 10, 2025.

Declarationen

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted hard copies.

---

place, date, signature