



HAL
open science

Dynamic Hierarchical Token Merging for Vision Transformers

Karim Haroun, Thibault Allenet, Karim Ben Chehida, Jean Martinet

► **To cite this version:**

Karim Haroun, Thibault Allenet, Karim Ben Chehida, Jean Martinet. Dynamic Hierarchical Token Merging for Vision Transformers. VISAPP-2025 - 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Feb 2025, Porto, Portugal. hal-04885469

HAL Id: hal-04885469

<https://hal.science/hal-04885469v1>

Submitted on 14 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Dynamic Hierarchical Token Merging for Vision Transformers

Karim Haroun^{1,2}^a, Thibault Allenet¹^b, Karim Ben Chehida¹^c, and Jean Martinet²^d

¹Université Paris-Saclay, CEA, List, F-91120 Palaiseau, France

²Université Côte d’Azur, I3S, CNRS, France

{karim.haroun, thibault.allenet karim.benchehida}@cea.fr, jean.martinet@univ-cotedazur.fr

Keywords: Vision Transformers, Token merging, Neural network compression, Dynamic neural networks

Abstract: Vision Transformers (ViTs) have achieved impressive results in computer vision, excelling in tasks such as image classification, segmentation, and object detection. However, their quadratic complexity $O(N^2)$, where N is the token sequence length, poses challenges when deployed on resource-limited devices. To address this issue, dynamic token merging has emerged as an effective strategy, progressively reducing the token count during inference to achieve computational savings. Some strategies consider all tokens in the sequence as merging candidates, without focusing on spatially close tokens. Other strategies either limit token merging to a local window, or constrains it to pairs of adjacent tokens, thus not capturing more complex feature relationships. In this paper, we propose Dynamic Hierarchical Token Merging (DHTM), a novel token merging approach, where we advocate that spatially close tokens share more information than distant tokens and consider all pairs of spatially close candidates instead of imposing fixed windows. Besides, our approach draws on the principles of Hierarchical Agglomerative Clustering (HAC), where we iteratively merge tokens in each layer, fusing a fixed number of selected neighbor token pairs based on their similarity. Our proposed approach is off-the-shelf, i.e., it does not require additional training. We evaluate our approach on the ImageNet-1K dataset for classification, achieving substantial computational savings while minimizing accuracy reduction, surpassing existing token merging methods.

1 INTRODUCTION

The advent of Vision Transformers (ViTs) (Dosovitskiy et al., 2020) has sparked significant advances in computer vision, demonstrating robust performance in image classification (Liu et al., 2021)(Touvron et al., 2021), segmentation (Zhang et al., 2022)(Strudel et al., 2021), and object detection tasks (Carion et al., 2020)(Liu et al., 2024). Since the introduction of the Vision Transformer (ViT), researchers have successfully adapted Transformers, originally designed for Natural Language Processing (NLP), to process images by treating local patches of an image as sequential tokens. Through the self-attention mechanism, ViTs learn the relationships between these tokens, achieving high-level visual understanding across a range of applications.

Despite these successes, ViTs have a notable limitation: their computational complexity scales

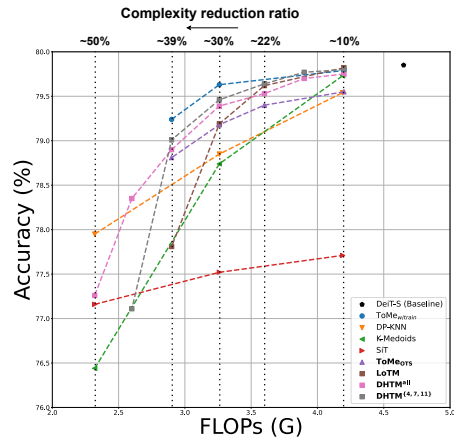






Figure 1: Performance of DHTM against existing methods on DeiT-Small (Touvron et al., 2021). We highlight in bold the off-the-shelf strategies, i.e., when no further training is required. We use the subscripts w/train and OTS for ToMe (Bolya et al., 2023) to distinguish between the off-the-shelf and the trained variants. The superscripts on DHTM denotes if the merging strategy is applied to all layers or empirically selected ones. The vertical dotted lines show complexity reduction ratio (FLOPs).

^a <https://orcid.org/0009-0000-6972-6019>

^b <https://orcid.org/0009-0003-0810-3338>

^c <https://orcid.org/0000-0002-5959-1832>

^d <https://orcid.org/0000-0001-8821-5556>

quadratically with respect to the number of tokens in the sequence. This $O(N^2)$ complexity, where N is the number of tokens, often limits their deployment on resource-constrained devices. Therefore, reducing the token sequence length has emerged as a practical strategy to make ViTs more computationally efficient, improving their adaptability to various hardware constraints while maintaining low accuracy reduction.

Among these, dynamic token reduction techniques are particularly prominent, encompassing two main strategies: token pruning and token merging. Token pruning selectively removes less significant tokens from the sequence, while token merging combines similar tokens, effectively fusing information and reducing redundancy. Both methods reduce complexity dynamically, i.e., at inference. This often results in a drop in accuracy, therefore, the main challenge is to find the optimal accuracy vs. complexity tradeoff. While token pruning effectively reduces computational load, it has two limitations. First, it risks losing crucial information, which may degrade model performance. Second, the variability in token importance across different inputs complicates batch processing. For these reasons, we focus on token merging approaches in this work.

In this paper, we propose Dynamic Hierarchical Token Merging (DHTM), a novel token merging strategy. Rather than selecting a single, fixed reference token and performing merging within a limited window, DHTM treats each token as a potential reference, iteratively expanding its region by merging with the most similar neighboring tokens in each Transformer layer. Our method is grounded on Hierarchical Agglomerative Clustering (HAC) (Ward Jr, 1963) and applies clustering in a localized manner, preserving essential information while minimizing information loss when merging tokens. By progressively merging tokens based on the highest similarities in each region, DHTM achieves efficient token reduction, significantly improving computational efficiency while minimizing accuracy reduction. As shown in Figure 1, DHTM effectively balances information loss with computational gains, offering a selective and thorough merging process that enhances model performance on ImageNet-1K dataset. The main contributions of this work are as follows:

- We introduce a DHTM, a spatially-aware token merging approach that iteratively combines similar neighboring tokens.
- We validate the effectiveness of our approach through extensive experiments on the ImageNet-1K dataset, showcasing that our method can reduce computational complexity while minimizing accuracy reduction.

- We evaluate the performance of DHTM against recent state-of-the-art token merging techniques, including global candidate evaluation and local window token merging strategies.
- We validate the merging criteria, i.e, the cosine similarity measure, through an ablation study against random merging.

2 RELATED WORKS

Vision Transformers (ViTs) traditionally process images by dividing them into a uniform grid of patches, with each patch treated as a token. However, not all regions of an image equally contribute to task performance, highlighting the need for efficient token management strategies. This section reviews Vision Transformers and state-of-the-art token merging techniques.

2.1 Vision Transformers

The flexibility of ViTs in handling variable-length inputs is a key feature that allows them to process multiscale visual inputs without requiring different sets of parameters. Each input image is divided into patches, projected to a latent space, and treated as tokens. While the number of tokens can vary depending on the image resolution or scale, the Transformer architecture is designed to handle this variability. The embedding size of each token is fixed, denoted as d_e , ensuring that each token is represented as a vector of the same dimensionality. Therefore, the input sequence of N tokens can be expressed as follows:

$$T = \{t_1, t_2, \dots, t_N\} \quad \text{where } t_i \in \mathbb{R}^{d_e} \quad (1)$$

Each token t_i is projected into three distinct representations: the *query* (Q), the *key* (K), and the *value* (V). The query Q encodes how much focus a token should receive, the key K encodes its relevance to other tokens, and the value V represents the content to be attended to. These projections are computed as $Q = TW_Q$, $K = TW_K$, and $V = TW_V$, where $W_Q, W_K, W_V \in \mathbb{R}^{d_e \times d_k}$ are learnable weight matrices, and d_k is the dimensionality of the query and key vectors.

The Multi-Head Self-Attention (MHSA) mechanism processes tokens pairwise by calculating attention scores for every token pair, as defined by the following equation:

$$\text{MHSA}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2)$$

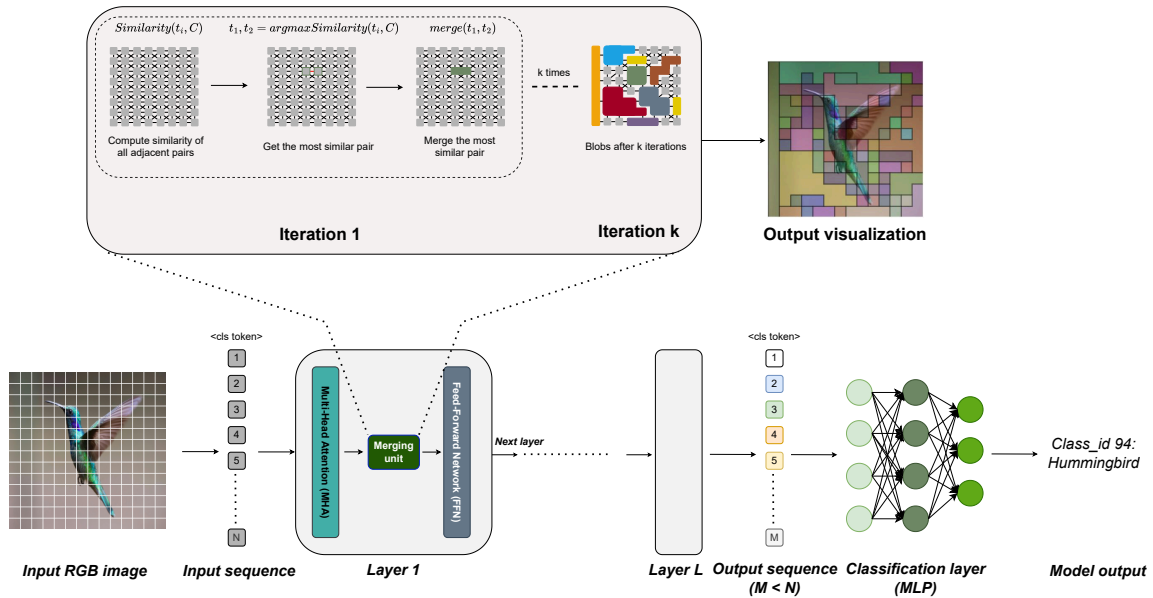


Figure 2: Overview of the DHTM method. Starting with an initial token set, we iteratively identify the most similar adjacent token pairs between t_i and its neighbors C . Then, we merge the most similar adjacent pair based on their cosine similarity, offering more flexibility than methods that restrict merging to local windows, while incorporating locality compared to approaches that merge distant tokens. Our strategy lies in the middle, leveraging both properties.

Similarly, the Multi-Layer Perceptron (MLP) layer that follows self-attention processes each token independently, applying the same transformation across all tokens due to their fixed embedding size. This design allows Transformers to generalize over varying input sizes while maintaining parameter efficiency. However, despite their advantages, ViTs exhibit quadratic complexity relative to the number of tokens, which increases the computational demand. The total number of Floating Point Operations (FLOPs) for a single Transformer layer can be expressed as follows:

$$\Phi_L(N, d_e) = \Phi_{\text{MHSA}}(N, d_e) + \Phi_{\text{MLP}} \quad (3)$$

$$= 12Nd_e^2 + 2N^2d_e \quad (4)$$

This quadratic complexity highlights the need for efficient token processing methods to mitigate the computational burden associated with larger input sizes. To address this challenge, researchers have proposed various token reduction strategies. The following subsection reviews state-of-the-art of token merging techniques, focusing on their strengths and limitations.

2.2 Token merging

Token merging combines tokens based on a similarity measure to improve efficiency. DPC-KNN

(Zeng et al., 2022) determines clusters by evaluating token density and merging those with minimal distance to higher-density points. SiT (Zong et al., 2022) and Sinkhorn (Haurum et al., 2022) use assignment matrices derived from learned queries to combine tokens. PatchMerger (Renggli et al., 2022) uses a dot-product softmax operation with preset queries for clustering. K-Medoids (Marin et al., 2023) applies a hard-clustering algorithm that iteratively minimizes Euclidean distances within clusters, using attention scores from the CLS token to initialize cluster centers. In ToMe (Bolya et al., 2023), tokens are split into two groups, with each token in one group paired and merged with its closest match in the other group by averaging their representations. Finally, LoTM (Haroun et al., 2024) constrains the merging to pairs of horizontally-adjacent tokens, relying on cosine similarity.

While these methods advance token merging, they have several limitations. PatchMerger (Renggli et al., 2022) and ToMe (Bolya et al., 2023) operate by evaluating all tokens globally as potential merging candidates, allowing distant clusters to be merged without emphasizing spatial relations, as spatially close tokens tend to share more semantic information than distant ones. Other approaches use a fixed local window around a predefined reference token, referred to as a centroid in these papers, to merge the most similar tokens within that window (Zeng

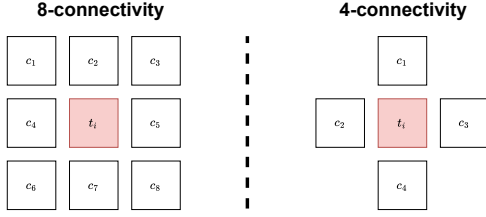


Figure 3: A 2D illustration of neighbor connectivity types. **Left:** 8-connectivity with all 8 neighbor tokens; **Right:** 4-connectivity, restricted to four horizontal and vertical neighbors. In DHTM, we use the 8-connectivity setting, as we test for all possible neighbors without restrictions.

et al., 2022)(Marin et al., 2023). Although promising, this constrained approach hinders the benefits of locality by setting rigid priors, such as reference tokens and the merging window. Lately, strategies like LoTM (Haroun et al., 2024) restrict merging candidates to two horizontally adjacent tokens, which is an extreme case in local merging between a pair of candidates. This restriction may hinder the method’s ability to capture more complex relationships and features among tokens. In contrast, our approach, considers all tokens as potential references and selectively merges only the most similar neighboring tokens in each Transformer layer, thus striking a balance between unrestricted and restrictive strategies.

3 METHODOLOGY

3.1 Token merging

Let $\mathbf{Z} \in \mathbb{R}^{N \times d_e}$ denote the output token sequence from the Multi-Head Self-Attention (MHSA) layer, depicted in Eq (2), where N is the number of tokens and d_e is the embedding dimension.

We define a similarity measure $S: \mathbb{R}^{d_e} \times \mathbb{R}^{d_e} \rightarrow \mathbb{R}$ for pairs of tokens. This similarity measure can be expressed as the inverse of the Euclidean distance, the inverse of the norm, or simply cosine similarity, although the latter is most commonly used.

Next, let $I \subseteq \{1, \dots, N\}$ represent a set of indices of tokens identified for merging based on a predefined threshold of similarity. The merging operation is defined as follows:

$$\mathbf{z}_{\text{merged}} = \text{avg_merge}(\{\mathbf{z}_i \mid i \in I\}) = \frac{1}{|I|} \sum_{i \in I} \mathbf{z}_i \quad (5)$$

where $\text{merge}(z_i, z_j)$ represents the average merge of z_i and z_j . After determining the merging results for selected pairs of tokens, the new token sequence can be expressed as follows:

Data: Set of tokens $T = \{t_1, t_2, \dots, t_N\}$,
Number of selected Transformer layers
 L , Number of merges per layer N_{merge}

Result: Final set of merged tokens T'

```

1 Initialize  $T' \leftarrow T$ ;
2 for each selected layer  $l = 1$  to  $L$  do
3    $M \leftarrow 0$ ; // Merge count
4    $\sigma \leftarrow []$ ; // Similarity score
5    $C \leftarrow []$ ; // Neighbors
6   while  $M < N_{\text{merge}}$  do
7     for each token  $t_i \in T'$  do
8        $C \leftarrow C + \text{get\_neighbors}(t_i, T')$ ;
9        $\sigma \leftarrow \sigma + \text{get\_similarities}(t_i, C)$ ;
10    end
11    /* Get the most similar pair */
12     $(t_1, t_2) \leftarrow C[\arg \max_k (\sigma[k])]$ ;
13     $t_m \leftarrow \text{avg\_merge}(t_1, t_2)$ ;
14     $T' \leftarrow (T' \setminus \{t_1, t_2\}) \cup \{t_m\}$ ;
15     $M \leftarrow M + 1$ ;
16  end
17 return  $T'$ ; //  $\text{len}(T') < \text{len}(T)$ 

```

Algorithm 1: DHTM algorithm

$$\mathbf{Z}' = \{\mathbf{z}_k \mid k \notin I\} \cup \{\mathbf{z}_{\text{merged}}\} \quad (6)$$

In this formalization, \mathbf{Z}' represents the updated token sequence after the merging operation, which reduces the sequence length.

3.2 Dynamic Hierarchical Token Merging (DHTM)

In this section, we provide additional information about the proposed DHTM strategy, which is described in Algorithm 1, in addition to the important steps. DHTM operates on an initial set of tokens $T = \{t_1, t_2, \dots, t_N\}$, obtained from the outputs of multi-head self-attention. The algorithm initializes a working set of tokens, denoted as T' , which serves as the basis for merging operations.

In each Transformer layer, DHTM first retrieves the spatial neighbors of each token $t_i \in T'$:

$$C = \text{get_neighbors}(t_i, T') \quad (7)$$

This operation ensures that the token merging process is local, focusing on nearby tokens, while testing all possible adjacent pairs of the sequence T' , without predefining hard merging windows or reference tokens, as shown in Figure 3 where we use the 8-connectivity setting for DHTM. Additionally, our

method also supports an alternative type of neighbor connectivity, which is the *4-connectivity*, where we constrain the connectivity to horizontal and vertical neighbors. After obtaining the neighbors, the algorithm evaluates the similarity between the token t_i and each of its neighbors $t_j \in C$ using a similarity measure:

$$\sigma = \text{get_similarities}(t_i, C) \quad (8)$$

This similarity measure quantifies how alike the tokens are, allowing for a more informed merging decision. While DHTM uses cosine similarity, other similarities or distance measures can be used, such as Euclidean distance, Manhattan distance or KL-divergence.

The algorithm then selects the neighbor with the highest similarity score and returns the best pair to merge:

$$(t_1, t_2) = C[\arg \max(S)] \quad (9)$$

This selection process ensures that, in each iteration, DHTM merges only the most similar token pairs, maintaining meaningful semantic information. Once the best neighbor is identified, the algorithm performs the average merging of the tokens:

$$t_m = \text{avg_merge}(t_1, t_2) \quad (10)$$

The merged token t_m then replaces the original tokens in T' , effectively reducing the sequence length. As shown in Figure 2, this merging process continues iteratively until the specified number of merges N_{merge} is reached for the current layer.

4 EXPERIMENTS

4.1 Dataset, Benchmarks and Comparison

We conduct our experiments on the ImageNet-1K dataset (Deng et al., 2009), a widely used benchmark for evaluating image classification models. ImageNet-1K contains over 1.2 million training images across 1,000 classes, with a validation set of 50,000 images, providing a diverse and comprehensive dataset to assess model performance.

To validate the effectiveness of our proposed Dynamic Hierarchical Token Merging (DHTM) method, we use the DeiT (Touvron et al., 2021) model as a backbone. Specifically, we evaluate our approach on three variants: DeiT-Tiny, DeiT-Small, and DeiT-Base. For comparison, we consider state-of-the-art token merging techniques on the same backbone.

4.2 Evaluation Metrics

The performance of our method is evaluated using two primary metrics: computational complexity and classification accuracy. We report the computational complexity in terms of Floating Point Operations (FLOPs), which is a standard measure for model efficiency, which we measure using the `fvcore`¹ library. We evaluate classification performance using the top-1 accuracy on the ImageNet-1K validation set, reflecting the percentage of correctly classified images.

4.3 Implementation details

As mentioned above, DHTM is designed to integrate into existing Transformer architectures without requiring additional training. During the evaluation, the batch size was set to 1. Additionally, in each selected layer, we iteratively merge k times, where k is predefined.

4.4 Experiment results

For a comprehensive comparison, we benchmark DHTM against several state-of-the-art token reduction methods, including SiT (Zong et al., 2022), Sinkhorn (Haurum et al., 2022), PatchMerger (Renggli et al., 2022), K-Medoids (Marin et al., 2023), DPC-KNN (Zeng et al., 2022), ToMe (Bolya et al., 2023), and LoTM (Haroun et al., 2024), based on the evaluation metrics depicted above.

Figure 1 depicts the performance of DHTM against state-of-the-art techniques on DeiT-Small (Touvron et al., 2021), given various computational budgets. We observe that for aggressive merging ratios exceeding 39%, DHTM^{all} performs better than DHTM^{4,7,11}, K-medoids, and SiT, while slightly trailing behind DP-KNN. Furthermore, DHTM demonstrates a key advantage by enabling higher compression ratios compared to other strategies, such as ToMe.

Since DHTM^{4,7,11} shows higher performance than DHTM^{all} for reduction ratios below 39%, we will use this configuration for the next experiments. Therefore, we will refer to it simply as DHTM for clarity.

Table 1 shows the performance of DHTM on three DeiT variants (Tiny, Small, and Base) with varying k values, illustrating the trade-off between accuracy and FLOPs. The results indicate that increasing k reduces FLOPs across models, with the largest reductions for higher k values, while the top-1 accu-

¹<https://github.com/facebookresearch/fvcore>

Table 1: DHTM performance comparison on DeiT (Touvron et al., 2021) models at different k values, where k denotes the number of merges applied in each selected layer. $k = 0$ represents the baseline, and $k = 56$ represents the most constrained configuration.

DeiT-Tiny		
k	Top-1(%)	FLOPs(G)
0	72.20	1.26
10	72.17	1.16
20	72.10	1.07
30	71.95	0.97
40	71.66	0.88
48	71.28	0.81
56	69.98	0.75

DeiT-Small		
k	Top-1(%)	FLOPs(G)
0	79.82	4.65
10	79.80	4.21
20	79.77	3.92
30	79.64	3.63
40	79.46	3.26
48	79.01	2.90
56	77.11	2.60

DeiT-Base		
k	Top-1(%)	FLOPs(G)
0	81.85	17.60
10	81.84	16.54
20	81.72	15.48
30	81.55	14.43
40	81.11	13.97
48	80.68	12.34
56	80.10	11.74

racy exhibits only a slight decline. Besides, we notice that the decrease in accuracy is less pronounced in the more complex DeiT-Base model, which has a larger embedding dimension ($d_e = 384$) twice that of DeiT-Small ($d_e = 768$) and four times that of DeiT-Tiny ($d_e = 192$). Larger embedding reduces the sensitivity to token merging, which allows more aggressive merging in DeiT-Base with minimal performance loss. This allows us to optimize computation efficiently while preserving accuracy, especially in larger models.

In table 2, we show the performance of DHTM against various state-of-the-art methods across three different models: DeiT-Tiny, DeiT-Small, and DeiT-Base. DHTM is an off-the-shelf method that requires no training and can be easily integrated into

Table 2: Performance evaluation of DHTM against existing methods for a reduction ratio of 30% in terms of FLOPs, we highlight Top-1 accuracy and FLOPs. W/o train means off-the-shelf variants, i.e., no additional training required, and w/train depicts variants that require training.

	Method	Top-1(%)	FLOPs(G)
w/ train	DeiT-Tiny	72.20	1.26
	Sinkhorn	53.19	0.88
	PatchMerger	66.81	-
	SiT	68.99	-
	DPC-KNN	70.10	-
	K-Medoids	69.90	-
	ToMe <i>w/train</i>	71.74	-
w/o train	ToMe <i>OTS</i>	70.94	-
	LoTM	70.76	-
	DHTM	71.66	-
w/ train	DeiT-Small	79.82	4.65
	Sinkhorn	64.02	3.26
	PatchMerger	75.80	-
	SiT	77.52	-
	K-Medoids	78.74	-
	DPC-KNN	78.85	-
	ToMe <i>w/train</i>	79.63	-
w/o train	ToMe <i>OTS</i>	79.18	-
	LoTM	79.19	-
	DHTM	79.46	-
w/ train	DeiT-Base	81.85	17.60
	Sinkhorn	63.36	12.34
	PatchMerger	74.52	-
	SiT	76.63	-
	DPC-KNN	79.06	-
	K-Medoids	79.98	-
	ToMe <i>w/train</i>	81.05	-
w/o train	ToMe <i>OTS</i>	80.75	-
	LoTM	80.01	-
	DHTM	80.68	-

any model, offering a plug-and-play solution, as are LoTM and ToMe_{OTS}. In contrast, ToMe_{w/train} (Bolya et al., 2023) requires training from scratch for 300 epochs, which increases computational cost. Finally, the results presented in the table are for $k = 40$, corresponding to a 30% reduction in complexity compared to the baseline. In the following paragraphs, we analyze the performance of our model on three DeiT variants compared to existing methods.

For the DeiT-Tiny model, the resource-constrained variant of DeiT, DHTM achieves 71.66% accuracy with 0.88G FLOPs at $k = 40$. DHTM outperforms Sinkhorn, PatchMerger, SiT, DPC-KNN, and K-Medoids by more than 1.5%. In addition, it slightly outperforms LoTM by 0.90%, indicating that restricting the merge to only two

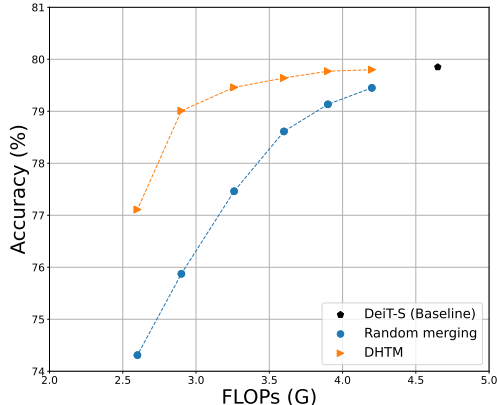


Figure 4: Comparison of DHTM with random token merging on DeiT-Small. The results clearly show that the random merging approach (in blue) experiences a significant drop in performance, particularly under more constrained scenarios as the number of merging candidates increases, while DHTM (in orange) maintains superior performance.

adjacent tokens may not fully capture the complexity of feature similarities. Finally, DHTM outperforms ToMe_{OTS} , an off-the-shelf variant of ToMe by 0.72%, and performs almost identically to $\text{ToMe}_{w/train}$, with a minimal difference of just 0.08%.

For the DeiT-Small model, DHTM achieves 79.46% accuracy with 3.26G FLOPs at $k = 40$. DHTM outperforms Sinkhorn, PatchMerger, SiT by 2% or higher, and K-Medoids, DPC-KNN by at least 0.61%. In addition, it slightly surpasses ToMe_{OTS} and LoTM by 0.28% and 0.27%, respectively, demonstrating the best performance for off-the-shelf models. Although it falls short by 0.17% compared to $\text{ToMe}_{w/train}$, this difference is minimal considering DHTM’s off-the-shelf deployment capability, unlike ToMe, which requires training.

Finally, DHTM achieves 80.68% accuracy with 12.34G FLOPs at $k = 40$ for DeiT-Base model. DHTM significantly outperforms Sinkhorn, PatchMerger, SiT, by more than 4.5%, and DP-KNN, K-Medoids by more than 1%. Furthermore, DHTM outperforms LoTM by 0.67%, but trails ToMe_{OTS} and $\text{ToMe}_{w/train}$ by 0.07% and 0.37% respectively.

These results demonstrate that DHTM outperforms most off-the-shelf methods except for ToMe on DeiT-Base and most training-dependent algorithms, where it trails slightly behind $\text{ToMe}_{w/train}$. This minor gap is offset by DHTM’s advantage of requiring no additional training.

4.5 Ablation study

To validate the merging decision in DHTM, we assess how well the cosine similarity-based merging compares to random merging of token candidates. Figure 4 demonstrates that our similarity-based token merging method preserves higher accuracy than random merging at equivalent FLOPs levels for both DHTM. Specifically, similarity-based merging consistently outperforms random merging, given all the configurations of k . For example, DHTM achieves 79.01% accuracy at 2.9 GFLOPs, whereas random merging achieves only 75.87%.

5 CONCLUSION

We introduced DHTM, an off-the-shelf dynamic token merging strategy that uses cosine similarity as the basis for merging decisions. Unlike existing methods that rely on a fixed centroid token, i.e., reference tokens to merge around, or constrain merging within a limited window, our approach iteratively aggregates spatially adjacent tokens by evaluating all neighbors of each token and selecting the most similar pair to merge in each step. This process progressively expands regions of similar tokens, effectively reducing computational overhead.

We designed the model based on two main intuitions: first, that merging decisions should prioritize spatially adjacent tokens, as these are more likely to convey similar information, corresponding visually to nearby patches. Second, our approach evaluates all spatially close token-merging candidates, iteratively selecting the most similar pairs. This removes the spatial restrictions imposed by some of the previous methods, allowing for a more flexible and comprehensive aggregation process, while relying on adjacent tokens. Our approach demonstrates minimal information loss compared to existing state-of-the-art methods on the ImageNet-1K dataset and achieves superior performance over most of these methods.

As a perspective, DHTM could be made even more flexible by enabling variable merging layerwise, with a threshold imposed on similarity instead of a fixed number of k merges per layer. Besides, the approach can be extended to dense prediction tasks, such as semantic segmentation and object detection, where token merging can enhance model efficiency and performance, particularly given the high complexity demands of these tasks.

REFERENCES

- Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. (2023). Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*.
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- Haroun, K., Martinet, J., Ben Chehida, K., and Allenet, T. (2024). Leveraging local similarity for token merging in Vision Transformers. In *ICONIP 2024 - 31th International Conference on Neural Information Processing*, Auckland, New Zealand.
- Haurum, J. B., Madadi, M., Escalera, S., and Moeslund, T. B. (2022). Multi-scale hybrid vision transformer and sinkhorn tokenizer for sewer defect classification. *Automation in Construction*, 144:104614.
- Liu, Y., Gehrig, M., Messikommer, N., Cannici, M., and Scaramuzza, D. (2024). Revisiting token pruning for object detection and instance segmentation. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2646–2656, Los Alamitos, CA, USA. IEEE Computer Society.
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.
- Marin, D., Chang, J.-H. R., Ranjan, A., Prabhu, A., Rastegari, M., and Tuzel, O. (2023). Token pooling in vision transformers for image classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 12–21.
- Renggli, C., Pinto, A. S., Houlsby, N., Mustafa, B., Puigcerver, J., and Riquelme, C. (2022). Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*.
- Strudel, R., Garcia, R., Laptev, I., and Schmid, C. (2021). Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272.
- Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.
- Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.
- Zeng, W., Jin, S., Liu, W., Qian, C., Luo, P., Ouyang, W., and Wang, X. (2022). Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11101–11111.
- Zhang, B., Tian, Z., Tang, Q., Chu, X., Wei, X., Shen, C., and Liu, Y. (2022). Segvit: Semantic segmentation with plain vision transformers. *NeurIPS*.
- Zong, Z., Li, K., Song, G., Wang, Y., Qiao, Y., Leng, B., and Liu, Y. (2022). Self-slimmed vision transformer. In *European Conference on Computer Vision*, pages 432–448. Springer.