Accelerating Transformers with Spectrum-Preserving Token Merging

Hoai-Chau Tran[1,2*], Duy M. H. Nguyen[1,3,4*], Duy M. Nguyen[5], TrungTin Nguyen[6], Ngan Le[7], Pengtao Xie[8,9], Daniel Sonntag[1,10], James Zou[11], Binh T. Nguyen[2], Mathias Niepert[3,4]

[1]German Research Center for Artificial Intelligence (DFKI), [2]University of Science - VNUHCM, [3]Max Planck Research School for Intelligent Systems (IMPRS-IS), [4]University of Stuttgart, [5]Dublin City University, [6]University of Queensland, [7]University of Arkansas, [8]MBZUAI, [9]UC San Diego, [10]Oldenburg University, [11]Stanford University.

Motivations

Vision Transformers (ViTs) excel in representing images and videos but have **high memory demands** and **quadratic time complexity** due to self-attention layers.

➔ model compression are required

Prior methods:

- Require retraining from scratch, limiting the use of pre-trained models.
- Adapting *Bipartite Soft Matching* (BSM) algorithms for token merging

➔ sensitive to token splitting strategies, potentially damaging informative tokens.

We propose ***PiToMe*** (Protect Informative Tokens before Merging), which can:

- Prioritize **preserving essential tokens** via "*energy score*".
- **Reduces FLOPs by 40–60% for ViT** with only a minimal performance drop.
- Theoretically **approximates the spectral structure** of the initial token space.

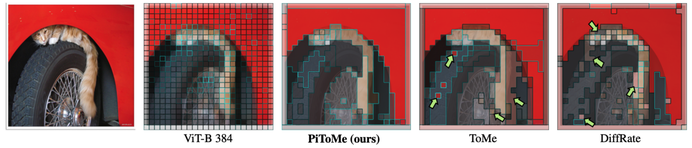


Figure 1. Comparison of token merging methods: same-color patches are merged. Green arrows mark errors avoided by PITOME. Tokens with high attention scores (cyan borders) retain positions proportional to ViT-base 384

PiToMe Algorithm

Token Graph Construction:

As in figure 2 **b)** first we need to construct a weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$, representing tokens connections. Here $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges each with a weights $\mathbf{W}[i,j]$, and $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix

$$\mathbf{W}[i,j] = 1 - \cos(v_i, v_j), \text{ where } \cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|}, \quad \forall v_i \in \mathcal{V}, v_j \in \mathcal{V}.$$

$\mathbf{W}[i,:]$ and $\mathbf{W}[:,i]$ denote the i-th row and column, *resp.*; $[N]$ stands for $\{1, \ldots, N\}$.

Token Energy Score

Let i be the index of the current node and $\mathcal{N}(i)$ represent the set of neighbor nodes. The energy score $E_i \equiv E_i(v_i, \mathbf{W}[i,:])$ of node v_i is calculated using the following equation:

$$E_i(v_i, \mathbf{W}[i,:]) = \frac{1}{N} \sum_{j \in \mathcal{N}(i)} f_m(\cos(v_i, v_j)), \; f_m(x) = \begin{cases} x & \text{if } x \geq m \\ \alpha(\exp(x-m) - 1) & \text{otherwise} \end{cases}.$$

➔ The energy score E reflect the density of neighbours tokens around each token. Tokens for large objects (e.g., background) have high scores are prime for merging, and the ones with lower score will be protected.

Token Merging With Energy Scores

Sorted E is then used to create two sets, $\mathcal{A}$ and $\mathcal{B}$ each with k nodes each. Tokens with odd indices go to $\mathcal{A}$, and those with even indices go to $\mathcal{B}$.

➔ Increasing the chance that tokens in $\mathcal{A}$ find their best matches in $\mathcal{B}$, unlike methods based on random spatial partitioning.

PiToMe Algorithm

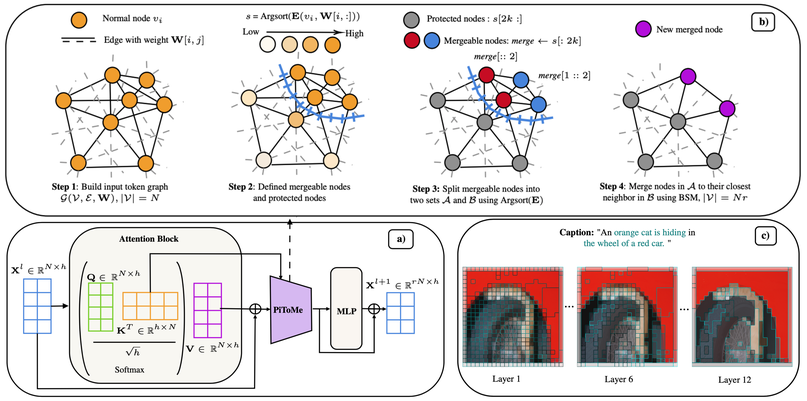


Figure 2. a) PITOME can be inserted inside transformer block; **b)** Energy scores are computed to identify mergeable and protective tokens; **c)** Our algorithm gradually merges tokens in each block.

Pseudo-Code Implementation

Algorithm 1 PiToMe Algorithm

function PITOME(remain token ratio: r, input graph: $\mathcal{G}(\mathcal{V}, \mathcal{E})$) // Function to prepare for merging
  $k \leftarrow N - N \cdot r$ // Compute number of nodes to merge
  $\mathbf{s} \leftarrow$ argsort($\mathbf{E}$, descending=True) // Compute energy scores
  $merge, protect \leftarrow \mathbf{s}[:2 \cdot k], \mathbf{s}[2 \cdot k:]$ // Identify mergeable and protected nodes
  $\mathbf{n}_a, \mathbf{n}_b \leftarrow merge[::2], merge[1::2]$ // Split mergeable nodes
  $\mathcal{E}_{merge} \leftarrow \mathcal{E}[\mathbf{n}_a][\mathbf{n}_b]$ // Get edge weights of mergeable nodes
  $\mathbf{n}_{dst} \leftarrow$ argmax$(\mathcal{E}_{merge})$ // Find closest neighbors
  function MERGE($\mathbf{X}$) // Function to perform merging
    $\mathbf{X}_{protected} \leftarrow \mathbf{X}[protect, :]$ // Extract protected tokens
    $\mathbf{X}_{\mathcal{A}}, \mathbf{X}_{\mathcal{B}} \leftarrow \mathbf{X}[\mathbf{n}_a, :], \mathbf{X}[\mathbf{n}_b, :]$ // Extract tokens in set $\mathcal{A}$ and $\mathcal{B}$
    $\mathbf{X}_{\mathcal{A}}, \mathbf{X}_{\mathcal{B}} \leftarrow \mathbf{X}_{\mathcal{A}} \times \mathbf{m}[\mathbf{n}_a], \mathbf{X}_{\mathcal{B}} \times \mathbf{m}[\mathbf{n}_b]$ // Weighted average
    $\mathbf{X}_{\mathcal{B}} \leftarrow \mathbf{X}_{\mathcal{B}}$.scatter_reduce($\mathbf{n}_{dst}, \mathbf{X}_{\mathcal{A}}$, mode = "sum") // Merge tokens
    $\mathbf{X}_{\mathcal{B}} \leftarrow \mathbf{X}_{\mathcal{B}} / \mathbf{m}_{\mathcal{B}}$.scatter_reduce($\mathbf{n}_{dst}, \mathbf{m}_{\mathcal{A}}$, mode = "sum") // Weighted average
    return cat($\mathbf{X}_{protected}, \mathbf{X}_{\mathcal{B}}$) // Concatenate and return merged tokens
  return MERGE // Return merging lambda function

- $\mathbf{E}$ is the vector that represent token energy scores.
- $\mathbf{m}$ is the vector that contain information about token size.
- MERGE is a lambda function that can be used for any matrix $\mathbf{X}$

Connection to Graph Coarsening with Spectral Preservation

Theorem 1 (Spectrum consistent of Token Merging). Suppose the graph $\mathcal{G}_0^{(s)}$, $\mathcal{G}_{\text{PiToMe}}^{(s)}$ and $\mathcal{G}_{\text{ToMe}}^{(s)}$ are coarsened from the original graph $\mathcal{G}$ by iteratively merging pairs of nodes v_{a_s} and v_{b_s} w.r.t the true partition $\mathcal{P}_0^{(s)} = \{\mathcal{V}_{0i}^{(s)}\}_{i \in [s]}$, the PiToMe-partition $\mathcal{P}_{\text{PiToMe}}^{(s)} = \{\mathcal{V}_{\text{PiToMe}i}^{(s)}\}_{i \in [s]}$, defined by PiToMe Algorithms 1, and the ToMe-partition $\mathcal{P}_{\text{ToMe}}^{(s)} = \{\mathcal{V}_{\text{ToMe}i}^{(s)}\}_{i \in [s]}$, for $s = N, \ldots, n+1$. We assume the following standard mild assumption:

(A1). $\mathbb{E}[\cos(v_{a_s}, v_{b_s})] \to 1, \quad \forall v_{a_s} \in \mathcal{V}_{0i}^{(s)}, \forall v_{b_s} \in \mathcal{V}_{0i}^{(s)}, i \in [s]$.

(A2). There exists a margin m s.t., $\cos(v_{a_s}, v_{b_s}) \geq m > \cos(v_{a_s}, v_{c_s}), \quad \forall v_{a_s} \in \mathcal{V}_{0i}^{(s)}, \forall v_{b_s} \in \mathcal{V}_{0i}^{(s)}, \forall v_{c_s} \in \mathcal{V}_{0j}^{(s)}, \forall i \neq j \in [s]$.

(A3). There is an order of cardinality in the true partition, without loss of generality, we assume $N_1^{(s)} \geq N_2^{(s)} \geq \ldots \geq N_s^{(s)}$, where $N_i^{(s)} = |\mathcal{V}_{0i}^{(s)}|, \forall i \in [s]$.

Then it hold that:

1. *The upper bound error $\epsilon_{\text{PiToMe}}^{(s)}$ between the edge weights of merged nodes from* PiToMe-*coarse $\mathcal{G}_{\text{PiToMe}}^{(n)}$ graphs converges to* 0, i.e.,
$$\epsilon_{\text{PiToMe}}^{(s)} \to 0 \text{ as } \mathbb{E}(\cos(v_{a_s}, v_{b_s})) \to 1, \text{for any } s = N, \ldots, n+1.$$

2. *The upper bound error $\epsilon_{\text{ToMe}}^{(s)}$ between the edge weights of merged nodes from ToMe-coarse $\mathcal{G}_{\text{ToMe}}^{(n)}$ graphs converges to a non-negative constant $C^{(s)}$, with a high probability that $C^{(s)} > 0$,* i.e.,
$$\epsilon_{\text{ToMe}}^{(s)} \to C^{(s)} \text{ and } \mathbb{P}(C^{(s)} > 0) > 0 \text{ as } \mathbb{E}(\cos(v_{a_s}, v_{b_s})) \to 1, \text{for any } s = N, \ldots, n+1.$$

Experimental Results

Table 1. Brief statistic of all datasets used in this paper.

Task	Name	Modality	No. Train	No. Test
Image-Text retrieval	Flickr30k [59]	Vision, Text	29k images, each with 5 captions	1k images, each with 5 captions
	MS-COCO[60]	Vision, Text	approx 118k images, each with 5 captions	5k images, each with 5 captions
Visual Question Answering	VQA-v2[69]	Vision, Text	approx 443.7k questions	approx 447.8k questions
	GQA [70]	Vision, Text	approx 72k images with 943k questions	approx 10.2k images with 12.5k questions
	VisWiz [71]	Vision, Text	8k image and questions	approx 4.32k image and questions
	TextVQA[73]	Vision, Text	approx 28.4k images with 34.6k questions	approx 3.1k images with 4.2k questions
	MME [74]	Vision, Text	-	2375 images and questions
	ScienceQA[72]	Vision, Text	approx 4.11k image and questions	approx 2.11k image and questions
Text Classification	IMDb [88]	Text	25k movie reviews.	25k movie reviews.
	SST-2 [87]	Text	67,349 sentences	872 sentences
Image Classification	Imagenet-1k[81]	Vision	Approximately approx 1.28 million images	50k images (50 images per class)

Backbones:

- **Image-Text retrieval:** CLIP, ALBEF, BLIP, BLIP2
- **VQA**: LLaVa-1.5-7B, LLaVa-1.5-13B.
- **Image classification:** DeiT, MAE.
- **Text classification:** BERT, DistilledBERT.

Baselines:

- ToMe, ToFu, DiffRate, DCT
- Others pretrained models with different architectures

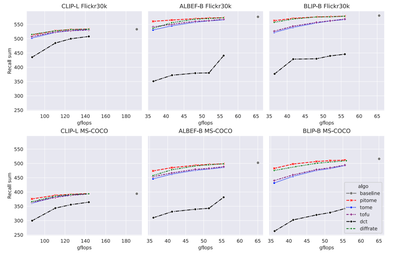
Experimental Results

✓ Image & Text Retrieval

Table 2. Off-the-shell and retrained PiToMe versus SOTA models

Datasets	Methods	Rt@1 ↑	Ri@1 ↑	ZS Retrieval Rsum ↑	Reranked Rsum ↑	ViT FLOPS ↓	Total FLOPS ↓	ZS Retrieval Time ↓	Total Time ↓
Flickr30k	ViLT	83.50	64.40	490.60	525.70	-	55.90	-	-
	LightingDOT	83.90	69.90	532.26	-	-	-	-	-
	UNITER	92.87	83.73	521.90	542.80	-	949.9	-	-
	METER	94.30	82.22	560.54	570.72	-	-	-	-
	CLIP-L	92.90	81.34	568.23	-	80.85	-	25s	-
	ALBEF	94.91	85.32	564.58	575.00	55.14	65.54	16s	58s
	$\text{PiToMe}^{\text{BLIP}}_{r=0.95}$	95.72	86.32	567.58	577.81	38.55	47.65	13s	56s
	$\text{PiToMe}^{\text{BLIP}}_{r=0.95}$	**96.61**	**87.18**	**569.98**	**579.35**	**38.55**	**47.65**	**13s**	**56s**
	BLIP	96.86	87.46	572.23	580.76	55.14	65.54	16s	1m11s
	$\text{PiToMe}^{\text{BLIP2}}_{r=0.95}$	96.83	87.84	566.25	580.77	296.93	390.77	45s	1m21s
	$\text{PiToMe}^{\text{BLIP2}}_{r=0.975}$	**97.55**	**89.04**	**572.81**	**583.72**	**434.50**	**564.78**	**1m5s**	**1m54s**
	BLIP2	97.61	89.79	572.72	584.76	678.45	900.77	1m37s	3m15s
MS-COCO	ViLT	61.50	42.70	420.20	439.20	-	55.90	-	-
	CLIP-L	70.78	53.79	478.18	-	80.85	-	2m10s	-
	METER	76.16	57.08	-	495.95	-	-	-	-
	ALBEF	76.94	60.24	478.39	500.44	55.14	65.54	43s	5m29s
	$\text{PiToMe}^{\text{BLIP}}_{r=0.95}$	79.46	62.50	485.99	506.65	38.85	47.65	51s	4m30s
	$\text{PiToMe}^{\text{BLIP}}_{r=0.95}$	**80.44**	**63.91**	**493.33**	**512.66**	**38.85**	**47.65**	**51s**	**4m30s**
	BLIP	81.82	64.36	494.34	516.03	55.14	65.54	1m3s	7m10s
	$\text{PiToMe}^{\text{BLIP2}}_{r=0.95}$	82.29	65.54	**494.92**	**518.44**	296.93	390.77	3m33s	6m34s
	$\text{PiToMe}^{\text{BLIP2}}_{r=0.975}$	**84.12**	**67.37**	**504.95**	**527.06**	**434.50**	**564.78**	**5m13s**	**9m24s**
	BLIP-2	85.32	68.26	507.46	528.63	678.45	900.77	7m52s	16m12s

Figure 3. Off-the-shell PiToMe versus baselines with different r



✓ Visual Question Answering (VQA) with Large Vision-Language Models

Table 3. Total inference time of LLaVa-1.5 after applying compressing methods

Model	VQA^{v2} ↓	GQA ↓	VisWiz ↓	ScienceQA^{I}↓	TextVQA↓	MME↓
LLava-1.5-7B	09h:05m	10m:25s	04m:36s	01m:50s	10m:12s	02m:32s
ToMe	05h:38m	06m:34s	03m:26s	01m:07s	07m:37s	01m:24s
ToFu	05h:35m	06m:32s	03m:29s	01m:06s	07m:40s	01m:24s
DCT	05h:59m	06m:41s	03m:28s	01m:08s	08m:16s	01m:27s
DiffRate	05h:39m	06m:39s	03m:26s	01m:06s	07m:36s	01m:21s
PiToMe	05h:44m	06m:37s	03m:26s	01m:07s	07m:37s	01m:23s
LLava-1.5-13B	13h:11m	13m:05s	07m:36s	04m:54s	15m:04s	02m:59s
ToMe	09h:28m	09m:35s	05m:58s	03m:31s	11m:48s	02m:16s
ToFu	09h:26m	09m32s	05m:58s	03m:26s	11m:45s	02m:15s
DCT	10h:02m	10m:53s	06m:46s	03m:45s	12m:57s	02m:34s
DiffRate	09h:33m	09:m44s	06m:01s	03m:37s	11m:52s	02m:18s
PiToMe	09h:32m	09m:39s	06m:03s	03m:35s	12m:08s	02m:17s

Figure 4. Off-the-shell PiToMe versus baselines with different r

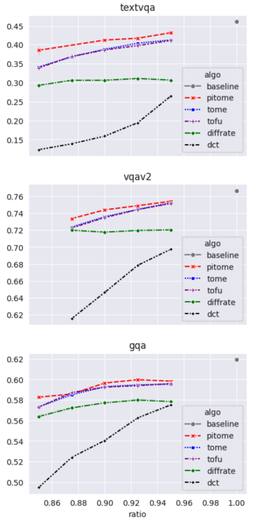
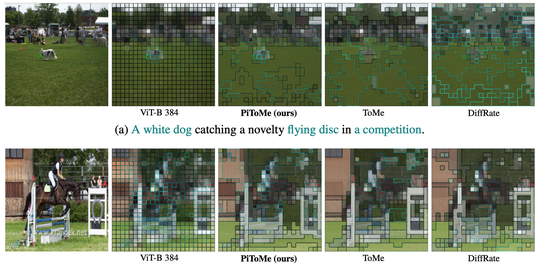


Figure 5. Visualizations using random images sampled from the MSCOCO dataset.

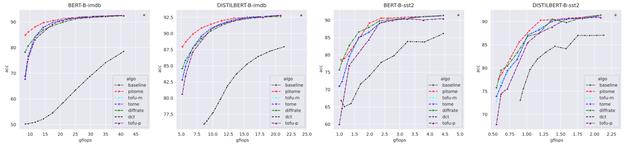


(a) A white dog catching a novelty flying disc in a competition.

(b) A woman riding a horse jumping it over obstacles.

✓ Text-Classification

Figure 6. Off-the-shell PiToMe versus baselines with different r



GitHub repo