TT-ACS: TRAINING-FREE TASK-AWARE ADAPTIVE COMPRESSION SCHEDULING FOR VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Token compression in vision transformers involves two distinct decisions: which tokens to compress and how much to compress. Existing methods primarily focus on how tokens are selected and compressed, while the compression budget is typically manually set and fixed across inputs. This input-agnostic design can under-compress redundant inputs while over-compressing information-rich ones. We introduce TT-ACS, a training-free framework for task-aware adaptive compression scheduling that determines the compression budget using task-relevant signals already computed by the model. Specifically, TT-ACS leverages attention interactions to estimate, for each input, the boundary between informative and redundant tokens, and maps this boundary to an adaptive compression budget. The resulting budget can be directly applied to existing token compression operators without modifying their internal mechanisms. On ImageNet-1K, TT-ACS consistently improves performance across all 13 operator–backbone configurations spanning four representative compression operators under matched compute budgets, with gains of up to 0.52 percentage points. Without task-specific retraining, TT-ACS also outperforms fixed-budget methods across multiple tasks, including object detection, video classification, and audio classification, with gains at all operating points on detection and video, and at light-to-moderate audio compression. These results demonstrate the value of compression scheduling as a distinct and complementary dimension of token compression, enabling compression strength to adapt to the task-relevant redundancy of each input. The code is available at https://anonymous.4open.science/r/TT-ACS/.

1 INTRODUCTION

Vision Transformers (Dosovitskiy et al., 2021) have taken the computer vision community by storm, achieving remarkable success across visual tasks driven by the spatial modeling mechanism of dot-product self-attention. However, this success comes at a computational cost that scales with the number of input tokens. Token compression is a natural response: pruning and merging operators (Rao et al., 2021; Liang et al., 2022; Bolya et al., 2023) discard or fuse redundant tokens to cut computation with little loss in accuracy, and among them, ToMe (Bolya et al., 2023) shows that such compression can even be applied to off-the-shelf pretrained models without any additional training.

These methods, however, mostly treat the compression budget as a manually specified global constant, applying the same compression degree to every input. This fixed budget creates a mismatch with input-dependent redundancy: for highly redundant inputs, it can be too conservative, leaving safe compression opportunities unused and wasting computation; for information-rich inputs, it can be too aggressive, increasing the risk of losing task-relevant information. This highlights two complementary dimensions of token compression: the compression operator determines how tokens are compressed, while the compression budget determines how much compression is applied. While existing operators provide effective ways to compress tokens, an adaptive budget can better match compression strength to each input. The key challenge is therefore to identify a signal that reads, for each input, how much compression the current task can tolerate.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

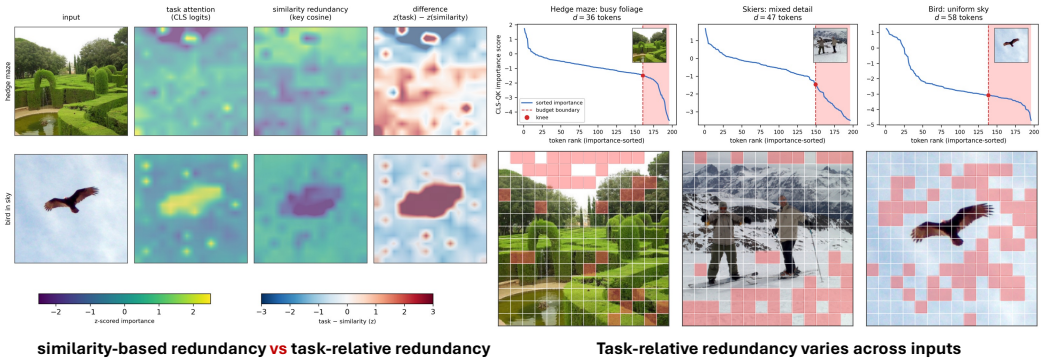

Figure 1: Similarity-based vs. task-relative redundancy: no single fixed budget matches all three inputs.

Existing token compression methods typically characterize redundancy by token similarity (Bolya et al., 2023), i.e., similarity between token representations. This provides a natural signal for identifying tokens that can be merged, but it does not directly indicate how much an input can be compressed without degrading task performance. We refer to this quantity as task-relative redundancy, defined as the amount of information that can be removed without compromising task performance. Estimating the compression budget therefore requires not a similarity signal, but a signal that reflects task relevance.

Attention provides a readily available proxy for task relevance (Abnar & Zuidema, 2020), and the resulting importance distribution exhibits a useful structure for estimating task-relative redundancy. In particular, the sorted importance profile often exhibits an informative head consisting of a small number of high-importance tokens, followed by a rapidly decaying tail of lower-importance tokens. The transition between these two regimes therefore provides a natural boundary for estimating the compression budget. As shown in Figure 1, importance is concentrated around task-relevant regions, and the score distribution reflects how task relevance is concentrated across tokens. This observation motivates using the head–tail boundary to estimate how much each input can be compressed.

Importantly, this does not make importance-based and similarity-based redundancy competing notions. They capture different aspects of redundancy and serve complementary purposes. Similarity tends to highlight background regions or areas with limited detail, making it effective for identifying tokens whose representations can be safely merged and thereby reducing the risk of information contamination during merging. In contrast, importance reflects the distribution of information relevant to the current task and determines the amount of compression that can be applied. Thus, importance guides the compression budget, while similarity guides the compression operation.

Based on this positioning, we propose TT-ACS, a training-free, task-aware compression scheduler that determines how much to compress from task-relevant attention signals, while remaining agnostic to how tokens are compressed. Unlike adaptive-compression methods (Fayyaz et al., 2022; Chen et al., 2023) that require training or modify the operator itself, TT-ACS leaves the compression operator untouched and takes over only the source of its budget, so it integrates with any off-the-shelf operator. TT-ACS can be directly combined with existing fixed-budget operators such as ToMe (Bolya et al., 2023). Our main contributions are as follows:

- We propose TT-ACS, a training-free, task-aware compression scheduler guided by task-relevant attention signals. TT-ACS is operator-agnostic and generalizes across tasks.
- On ImageNet-1K, integrating TT-ACS with four representative compression operators consistently improves accuracy across all 13 operator–backbone configurations under matched computational budgets. Without task-specific retraining, TT-ACS also outperforms fixed-budget baselines on object detection, video classification, and audio classification, with gains across all operating points for detection and video classification, and under light-to-moderate compression for audio classification.

2 Related Works

Token compression methods make two orthogonal decisions: the compression operator, which decides which tokens to prune or merge and how to aggregate them, and the compression budget, which decides how many tokens each input can afford to lose. Existing research has focused almost entirely on the operator, while the budget has received far less attention and, where it is specified at all, is a constant fixed by hand.

2.1 Fixed-Budget Pruning and Merging

Pruning and merging operators differ in how they select and aggregate tokens, yet across this literature the budget is fixed before inference. DynamicViT (Rao et al., 2021), EViT (Liang et al., 2022), ToMe (Bolya et al., 2023), MPM (Ravé et al., 2026), MaMe (Huo & Li, 2026), and ToMeSD (Bolya & Hoffman, 2023) operate with per-layer merge or drop counts specified in advance. PiToMe (Tran et al., 2024), FastV (Chen et al., 2024), and PyramidDrop (Xing et al., 2025) set a global retention ratio. SAINT (Jeddi et al., 2025) fixes a similarity threshold. The form varies, but its value never depends on the input. This input-independence, not any particular setting, is what makes the budget a separate problem: one constant must serve every input, so it must be conservative for information-rich inputs, leaving safe compression unused on redundant ones. TT-ACS targets this lever and leaves the operator untouched.

2.2 Adaptive Budget Allocation

A smaller line of work makes the budget itself adaptive, and the existing attempts differ in where the adaptivity resides. DiffRate (Chen et al., 2023) performs a differentiable search over per-layer prune and merge rates, which requires training. ATS (Fayyaz et al., 2022) selects tokens according to attention-based significance scores, yet its per-layer sampling target remains a preset constant. ATM (Lee & Choi, 2025) calibrates similarity thresholds without training, but the thresholds reside inside the merging operator. AdaMerge (Lee et al., 2026) adapts per-layer compression intensity from token salience, again with the mechanism fused into the operator. RAPID (Choi & Jang, 2026) is training-free but relies on a manually specified split between pruning and merging stages across depth. These methods establish that the budget need not be a constant; what they share is that the budget decision remains fused with the operator or is set by hand, so it cannot be scheduled on top of an untouched off-the-shelf compressor.

This coupling is not specific to token compression. Dynamic networks adjust computation per input through early exit, conditional computation, and dynamic depth (Han et al., 2022), but they do so by modifying the inference path or training additional routers. In object detection, Sparse DETR (Roh et al., 2022) and Focus-DETR (Zheng et al., 2023) retain a fixed token fraction selected by a learned scorer, so the budget is again preset and the scorer must be trained. Across architectures, from ViT classification to diffusion models (Zhang et al., 2025) and multimodal LLMs (Chen et al., 2024; Xing et al., 2025), the budget remains preset.

2.3 Attention as a Task-Relevance Signal

Deciding how much to compress requires a per-input signal for how much compression the task can bear. Attention is a natural candidate: it introduces no additional computational overhead at inference time and is oriented toward the prediction objective (Abnar & Zuidema, 2020). Attention-based importance already underpins several compression methods. EViT (Liang et al., 2022) ranks tokens by CLS attention and prunes the lowest-ranked tail under a fixed keep count; ATS (Fayyaz et al., 2022) samples tokens by attention-based significance scores under a preset per-layer sampling target; FastV (Chen et al., 2024) discards low-attention tokens in multimodal LLMs. In all of these methods, attention decides which tokens to discard, while the amount discarded remains preset. Using attention to decide how much to compress remains unexplored, and this is the role that TT-ACS assigns to the signal. This positioning separates TT-ACS from its nearest neighbors: unlike EViT (Liang et al., 2022), FastV (Chen et al., 2024), and ATS (Fayyaz et al., 2022), TT-ACS uses the attention signal to set the budget rather than the ranking; unlike ATM (Lee & Choi, 2025) and AdaMerge (Lee et al., 2026), its scheduler is external to the operator; and unlike DiffRate (Chen et al., 2023), it requires no training.

3 METHOD

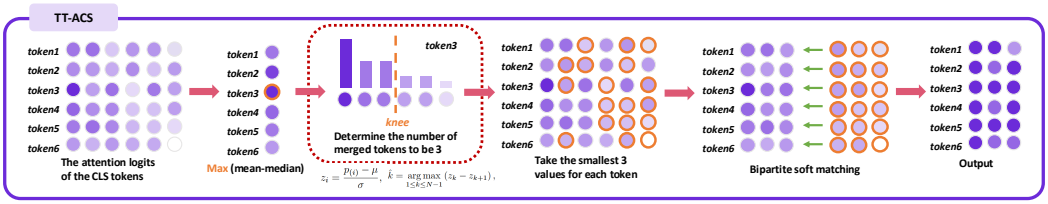

Figure 2: **Overview of TT-ACS.** From the frozen model's pre-softmax CLS-row logits, TT-ACS derives per-token importance scores. Conservative boundary estimation separates the informative head from the redundant tail of the sorted scores. The resulting budget is scheduled across layers and executed by the unchanged ToMe operator.

TT-ACS reformulates token compression as an input-adaptive budget scheduling problem: it estimates a budget $r_x = f(x)$ directly from the input under frozen pretrained weights, then hands it to a merging operator for execution, without introducing learnable parameters or fine-tuning. Because the two decisions are orthogonal, any operator that accepts a budget can serve as the executor. Figure 2 summarizes the pipeline: (i) task-aware token importance estimation (Section 3.1) aggregates the attention logits already computed during inference into a one-dimensional importance profile; (ii) conservative boundary estimation (Section 3.2) locates the boundary between the informative head and the redundant tail of the sorted distribution, yielding a per-input budget; (iii) cross-layer scheduling and execution (Section 3.3) turns this budget into per-layer merge counts under the scheduling mode specified by the task, and hands them to the operator.

3.1 TASK-AWARE TOKEN IMPORTANCE ESTIMATION

An adaptive budget requires a signal reflecting how relevant each token is to the current task, derived from the prediction process itself. To this end, we use the pre-softmax QK logits that the model already computes during inference:

$$\alpha^{\ell}_{h,q,i} = \frac{1}{\sqrt{d_h}} \left(\mathbf{q}^{\ell}_{h,q}\right)^{\top} \mathbf{k}^{\ell}_{h,i}, \tag{1}$$

where h indexes attention heads, q indexes queries, i indexes attended tokens, and d_h is the head dimension. We use raw logits rather than softmax-normalized attention probabilities: softmax suppresses the low-value tail and compresses the contrast between informative and redundant tokens, whereas boundary detection relies precisely on this contrast.

Different tasks expose their relevant interactions at different query positions, so the aggregation query set should be task-specific. We therefore aggregate multi-head responses into a one-dimensional importance profile at layer ℓ over a task-specific query set $\mathcal{Q}$:

$$p^{\ell}_i = \frac{1}{H|\mathcal{Q}|} \sum_{h=1}^{H} \sum_{q \in \mathcal{Q}} \alpha^{\ell}_{h,q,i}, \tag{2}$$

where H is the number of heads. Averaging over heads and queries collapses the $H|\mathcal{Q}|$ per-head, per-query importance lists into a single stable profile, the input consumed by the boundary estimator of Section 3.2. A larger p^{ℓ}_i indicates stronger task relevance, while tokens with low p^{ℓ}_i are likely redundant. Across classification, detection, video, and audio tasks, the same aggregation (Eq. (2)) supplies the budget signal; adapting to a new task primarily requires changing the source of $\mathcal{Q}$.

3.2 CONSERVATIVE BOUNDARY ESTIMATION

Fix a decision layer ℓ and write p_i for p^{ℓ}_i. The key question is how many of the N tokens should be merged for a given input, and every design choice in this subsection is governed by a single principle: the two error modes are asymmetric. Over-compression irreversibly destroys task-relevant information, whereas under-compression only wastes some computation; the estimator should therefore err toward keeping more tokens.

At the layers where the decision is taken, task-relevant attention exhibits a sharp head of a few critical tokens followed by a rapidly decaying tail, and the transition between the two is a natural compression boundary: tokens in the head are kept, and the tail defines the budget. We sort the scores in descending order, $p_{(1)} \geq \cdots \geq p_{(N)}$, and take the largest adjacent drop as the primary estimate of this transition:

$$\hat{k} = \underset{1 \leq k \leq N-1}{\arg\max} \left(p_{(k)} - p_{(k+1)} \right), \tag{3}$$

where k is the number of tokens in the informative head. This estimator presumes that a head-to-tail structure exists at the decision layer. Shallow layers do not meet this condition, as their sorted profiles are nearly flat; the decision position is therefore a per-task structural choice rather than a free parameter.

The max-gap estimate has one failure mode (Figure 3): the largest drop can fall within a high-value plateau, a stretch of nearly equal, near-top scores that reflects the internal structure of the informative head rather than the head-to-tail boundary. Merging according to such a knee destroys information, an effect we isolate experimentally in Section 4.2.

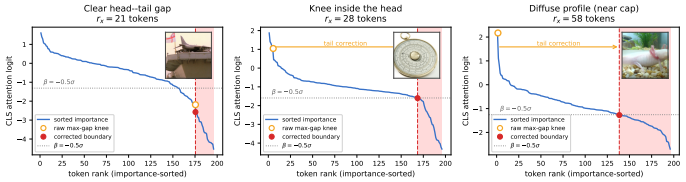

Figure 3: Tail correction on three real CLS-attention profiles (orange: raw knee; red: corrected boundary).

We further bound the kept set from below by a conservative threshold on the Z-standardized profile, $z_{(i)} = (p_{(i)} - \mu)/\sigma$, where μ and σ are the mean and standard deviation of the sorted profile:

$$k_\beta = \min\big(\{k : z_{(k)} < \beta\} \cup \{N\}\big), \tag{4}$$

the first position whose Z-score falls below β, or the sequence end N if no position does. The corrected boundary is

$$k^* = \max\big(\hat{k},\, k_\beta\big). \tag{5}$$

Because the sorted Z-scores are non-increasing, Eq. (5) unifies three regimes. If the knee lies strictly inside the head ($z_{(\hat{k})} > \beta$), every position up to it scores above the threshold, so the threshold position lies after the knee and the boundary moves tailward; if the knee already lies below the threshold, it is itself the more conservative of the two candidates; and if no position falls below the threshold, $k_\beta = N$ and no compression is applied, which also guards against over-merging on near-flat profiles. In every regime the corrected boundary keeps at least as many tokens as the raw knee: the correction acts directionally rather than at a precise cutoff, which is why the rule is robust to the exact value of β.

Z-standardization leaves the knee position unchanged, since the argmax in Eq. (3) is invariant to positive affine transformations of p; its role is to place β on a scale comparable across layers and images, so that a single threshold $\beta = -0.5$ is reused at every decision point and task.

The instance-level budget is then

$$r_x = N - k^*, \qquad \rho_x = \min\left(\frac{r_x}{N},\, \rho_{\max}\right), \tag{6}$$

where the cap $\rho_{\max}$ bounds the one failure mode that the conservative mechanism cannot exclude by construction: on a near-flat profile the largest adjacent drop can be a spurious transition inside the tail, and the cap limits the damage of such events. Its scope depends on the scheduling mode: in per-layer re-decision scheduling it caps each layer's decision, whereas in decide-once scheduling it caps the total budget before the split across the execution window (Algorithm 2). Eqs. (1) through (6) fully specify the estimator $f(x)$.

3.3 Cross-Layer Scheduling and Execution

A per-input budget does not yet specify what happens at each layer; the scheduler maps the budget onto the depth axis. Two modes cover the tasks we consider. In decide-once scheduling, the budget is read from the attention at a single decision layer D and spread uniformly over the execution window $[a, b]$ of layers after D, so the scheduled merges are distributed across depth rather than spent at a single position; detection and video follow this mode. In per-layer re-decision scheduling, the boundary is re-estimated from the attention at each layer in the window, so the merge count tracks the layer-wise decay of redundancy (Figure 4); classification and audio follow this mode. The choice of mode is part of the per-task instantiation rather than a tunable preference. Throughout, layers are indexed from 0 ($0, \ldots, 11$ for a 12-layer backbone). The concrete instantiations, including decision positions and execution windows, are given in Appendix A.1.

4 Experiments

In this section, we validate TT-ACS in two respects: effectiveness and robustness. Section 4.1 integrates TT-ACS with four representative operators on classification and transfers it to detection, video, and audio, verifying accuracy gains and cross-task generality; Section 4.2 ablates the boundary-detection algorithm and decision window and examines sensitivity to the correction threshold and batch size; Section 4.3 presents a visual comparison of fixed and TT-ACS adaptive budgets across the four tasks. Experimental configurations and details are given in Appendix A.1.

Evaluation protocol. We compare against each operator's own fixed-budget envelope: for each operator–backbone pair, we sweep its original budget parameters to obtain an accuracy–compute curve, and linearly interpolate this envelope at the GFLOPs (or executed compression ρ) actually measured for TT-ACS, against which the gains are computed (ΔAcc in Table 1, and ΔAP@ρ, Δtop-1@GFLOPs, and ΔmAP@ρ in Table 2), so that the comparison anchors remain consistent and verifiable. ΔFLOPs is the relative GFLOPs change of +TT-ACS with respect to the operator alone; all remaining Δ columns are absolute differences. All im/s numbers are the mean $\pm$ std over three repeated measurements. Within the Parameters group, the Method column lists the operator's own budget parameter; the TT-ACS column lists the decision window $[a, b]$ and cap $\rho_{\max}$ of TT-ACS.

4.1 Main Results

		Parameters		Operator alone				+ TT-ACS		
Operator	Backbone	Method	TT-ACS	Acc	Matched Acc	GFLOPs	im/s	Acc (ΔAcc)	GFLOPs (ΔFLOPs)	im/s (Δim/s)
ToMe	DeiT-T/16	r^{ToMe}=8	[7, 11], $\rho_{\max}$=0.5	71.43	71.33	0.93	991±61	**71.54 (+0.21)**	0.91±0.00 (−2.2%)	1125±19 (+134)
	DeiT-S/16	r^{ToMe}=12	[5, 11], $\rho_{\max}$=0.5	78.92	78.46	2.85	1010±2	**78.97 (+0.51)**	2.67±0.05 (−6.3%)	1010±5 (+0)
	DeiT-B/16	r^{ToMe}=12	[5, 11], $\rho_{\max}$=0.5	80.12	78.89	10.91	1005±1	**79.38 (+0.49)**	9.87±0.05 (−9.6%)	811±2 (−194)
	ViT-S/16	r^{ToMe}=4	[9, 11], $\rho_{\max}$=0.5	74.70	74.70	4.01	1049±6	**74.91 (+0.21)**	4.18±0.03 (+4.2%)	1381±5 (+332)
	ViT-B/16	r^{ToMe}=6	[8, 11], $\rho_{\max}$=0.5	80.72	80.77	14.21	889±2	**80.94 (+0.17)**	14.44±0.15 (+1.6%)	862±8 (−27)
	CLIP ViT-B/16	r^{ToMe}=8	[7, 11], $\rho_{\max}$=0.5	84.77	84.71	13.11	939±0	**84.92 (+0.21)**	12.80±0.14 (−2.4%)	830±9 (−109)
	MAE ViT-B/16	r^{ToMe}=4	[9, 11], $\rho_{\max}$=0.5	83.34	83.34	15.33	837±19	**83.43 (+0.09)**	15.81±0.82 (+3.2%)	858±23 (+21)
PiToMe	DeiT-S/16	ratio 0.9125	[0, 11], $\rho_{\max}$=0.1	78.09	78.31	2.78	748±2	**78.63 (+0.32)**	2.88±0.12 (+3.8%)	500±7 (−248)
	DeiT-B/16	ratio 0.9125	[0, 11], $\rho_{\max}$=0.1	79.95	79.95	10.63	695±9	**80.46 (+0.51)**	10.64±0.51 (+0.1%)	465±10 (−231)
EViT	DeiT-S/16	keep 0.70	$\{3, 6, 9\}$, $\rho_{\max}$=0.35	78.43	78.43	3.03	2004±18	**78.95 (+0.52)**	2.89±0.12 (−4.5%)	1473±34 (−531)
	DeiT-B/16	keep 0.70	$\{3, 6, 9\}$, $\rho_{\max}$=0.35	80.31	80.31	11.61	1160±21	**80.68 (+0.37)**	11.22±0.94 (−3.4%)	1019±9 (−142)
SAINT	DeiT-S/16	s=0.90	[9, 11], $\rho_{\max}$=0.1	79.66	79.61	4.60±0.01	1678±27	**79.70 (+0.09)**	4.45±0.03 (−3.3%)	1296±4 (−383)
	DeiT-B/16	s=0.85	[9, 11], $\rho_{\max}$=0.1	81.35	81.38	16.54±0.36	878±3	**81.70 (+0.32)**	16.91±0.02 (+2.2%)	807±3 (−72)

Table 1: Integrating TT-ACS with four operators on ImageNet-1K val. Matched Acc is the operator's accuracy linearly interpolated at the GFLOPs measured for +TT-ACS. In the Method column, r^{ToMe} is ToMe's per-layer merge count, ratio/keep are the PiToMe/EViT global retention rates, and s is SAINT's similarity threshold; the EViT rows re-decide at their three pruning layers $\{3, 6, 9\}$.

As shown in Table 1, we integrate TT-ACS with four representative operators. At matched compute, all operator–backbone configurations show consistent accuracy improvements of +0.09 to +0.52 percentage points. Notably, the gains are independent of the budget form: whether it is a per-layer count or a global ratio, the fixed-budget mismatch of being too conservative for redundant inputs and too aggressive for information-rich ones genuinely exists, and replacing the budget source alone mitigates it, confirming the budget as a lever independent of the operator. Since the budget decisions

introduce extra computation and the adaptive decisions are themselves conservative, the ΔFLOPs range from -9.6% to $+4.2\%$, and the throughput fluctuates accordingly. We additionally compare TT-ACS on these four operators against existing budget methods (Appendix A.2).

COCO object detection (DETR-R50)

Parameters		ToMe alone				+ TT-ACS				
Method	TT-ACS	AP	AP50/AP75	GFLOPs	ms	AP (ρ)	AP50/AP75	GFLOPs (Δ)	ms (Δ)	ΔAP@ρ
r^{ToMe}=8	D=1, [2, 5], $\rho_{\max}$=0.11	41.25	62.01/43.53	153.48	48.7	**41.90 (10.5%)**	62.32/44.28	154.10 (+0.62)	53.9 (+5.2)	+0.65
r^{ToMe}=16	D=1, [2, 5], $\rho_{\max}$=0.22	41.01	61.88/43.30	152.80	51.9	**41.64 (13.1%)**	62.03/44.12	153.79 (+0.99)	53.5 (+1.6)	+0.63
r^{ToMe}=48	D=2, [3, 5], $\rho_{\max}$=0.33	39.14	60.71/40.98	150.16	52.2	**40.70 (30.8%)**	61.61/43.81	152.51 (+2.35)	54.8 (+2.6)	+1.56

Kinetics-400 video classification (VideoMAE ViT-B)

Parameters		ToMe alone			+ TT-ACS			
Method	TT-ACS	top-1/top-5	GFLOPs	clips/s	top-1/top-5 (ρ)	GFLOPs (Δ)	clips/s (Δ)	Δtop-1@GFLOPs
r^{ToMe}=16	D=1, [2, 11]	74.80 / 93.13	167.5	96.0	**75.00 / 93.20 (2.8%)**	176.7 (+9.2)	89.6 (-6.4)	+0.20
r^{ToMe}=16	D=2, [3, 11], $\rho_{\max}$=0.15	74.35 / 93.19	167.5	96.0	**74.90 / 93.10 (15.0%)**	168.5 (+1.0)	94.4 (-1.6)	+0.55
r^{ToMe}=32	D=2, [3, 11], $\rho_{\max}$=0.3†	74.35 / 93.15	154.6	106.4	**75.00 / 93.20 (24.1%)**	160.6 (+6.0)	102.5 (-3.9)	+0.65

AudioSet audio classification (Audio MAE ViT-B)

Parameters		ToMe alone		+ TT-ACS		
Method	TT-ACS	mAP	samples/s	mAP (ρ)	samples/s (Δ)	ΔmAP@ρ
r^{ToMe}=10	D=9, [10, 11]	46.13	387.6	**46.26 (15.7%)**	321.5 (-66.1)	+0.13
r^{ToMe}=20	D=6, [7, 11]	45.52	454.0	**45.97 (39.1%)**	343.5 (-110.5)	+0.45
r^{ToMe}=30	D=3, [4, 11]	44.64	534.1	44.50 (60.4%)	375.8 (-158.3)	-0.14

Table 2: Integrating TT-ACS with ToMe and transferring it to detection (COCO), video (Kinetics-400), and audio (AudioSet). D denotes the decision layer: the budget is read from the attention at layer D; $[a, b]$ denotes the execution window, i.e., the layers on which merging is executed following the decision at D. The ρ reported for +TT-ACS is the mean executed compression over the evaluation set. †: $\rho_{\max}$ does not bind; the executed compression is set by the knee.

Table 2 transfers TT-ACS (with ToMe as the operator) to three tasks, requiring only a task-specific attention-signal source and no training. At matched compression (or compute), all three detection operating points exceed the fixed-budget envelope ($+0.65$/$+0.63$/$+1.56$ AP), and all three video operating points are likewise positive ($+0.20$/$+0.55$/$+0.65$ top-1); audio is positive at the light and moderate tiers ($+0.13$/$+0.45$ mAP) and slightly below the envelope at the deepest tier (-0.14). To reach the executed compression of the deepest tier, we moved the decision window earlier, to layers where the head–tail structure is inherently weaker, degrading the signal quality and enlarging the decision error of the adaptive budget. We further observe in the first detection row that TT-ACS executes 10.5% compression, which is more aggressive than in the nearest fixed-budget configuration (5.4%), yet achieves higher AP (41.90, above both that lighter configuration's 41.61 and the envelope of 41.25 at matched compression). This directly shows the essential difference between adaptive and fixed budgets: on input distributions with more redundancy it compresses more aggressively, whereas the fixed budget, having to serve every input, is forced to be conservative. The span of the executed compression ρ across the three tasks (10.5%–60.4%) is itself evidence that the executed compression varies automatically with the task and the input: the budget is decided by the knee, not by a manual tier.

4.2 Ablation Analysis

Decision algorithm. As shown in Table 3(a), under the same signal and the same window, Z-score max-gap performs best, while the six alternative change-point detectors all lose 0.28 percentage points, with $\rho_{\max}$ hit rates of 38%–51%. In comparison, the hit rate of our rule is only 6%. The $\rho_{\max}$ hit column explains why the alternatives' accuracies are identical: they cannot read out the head–tail transition from the sorted distribution, so a large fraction of per-layer decisions saturate the per-layer cap and the decision effectively degenerates into a fixed budget. In other words, what fails is not the idea of boundary detection but detectors lacking the ability to read out the head–tail structure; the accuracy gains are indeed attributable to the max-gap rule with tail correction itself. Appendix A.3 further validates the attention-derived budget against occlusion importance and merge interventions.

(a) Decision algorithm

Method	Acc	ΔAcc	GFLOPs	$\rho_{\max}$ hit	Throughput
Z-score	**79.626**	—	3.560	6%	1137
Pettitt (Pettitt, 1979)	79.344	−0.28	3.200	46%	899
Mann–Kendall (Mann, 1945)	79.344	−0.28	3.200	51%	907
BCPD (Adams & MacKay, 2007)	79.344	−0.28	3.223	38%	820
CUSUM (Page, 1954)	79.344	−0.28	3.200	46%	969
GMM	79.344	−0.28	3.208	42%	307
Otsu (Otsu, 1979)	79.344	−0.28	3.200	39%	580

(b) Decision window

Model	Acc	GFLOPs	ΔFLOPs	Throughput
Baseline	79.724	4.599	—	2273
ToMe (r^{ToMe}=8)	79.318	3.421	−25.6%	1290
TT-ACS [9, 11]	**79.744**	4.130	−10.2%	1358
TT-ACS [7, 11]	79.626	3.560	−22.6%	1137
TT-ACS [6, 11]	79.288	3.205	−30.3%	1217
TT-ACS [3, 11]	79.364	2.701	−41.3%	816

Table 3: Component ablations on DeiT-S/16 (ImageNet-1K val), where $\rho_{\max}$ hit is the fraction of layer decisions saturating the upper bound $\rho_{\max}$.

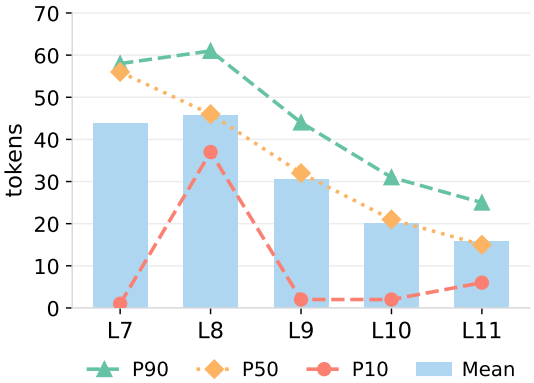

Figure 4: Layer-wise per-image max-gap knee budgets on ImageNet-1K val images (DeiT-S/16).

Decision window. As shown in Figure 4, we observe three regularities. First, the mergeable budget shrinks toward deeper layers: in the last layers what remains is mostly critical information, which should not be compressed further. Second, the ℓ=7 distribution is strongly bimodal: some images have almost no early-layer redundancy while others need heavy early merging, so a fixed budget would either leave real redundancy unmerged or destroy information where there is none; this is precisely why the budget must be adaptive. Third, although single-layer budgets fluctuate strongly across images, the per-image total is comparatively stable and concentrated: a roughly fixed total budget with dynamically allocated per-layer shares. The per-input budget is scheduled over depth rather than spent at a single layer: the earlier merging happens, the more FLOPs are saved because all subsequent layers operate on shorter sequences, but the perturbation to the representation also occurs earlier and propagates further, so the window position is a direct accuracy–efficiency trade-off. Table 3(b) also shows that the window position yields a smooth trade-off rather than a knife-edge one: the latest window $[9, 11]$ loses almost no accuracy while FLOPs drop by only 10.2%; extending the window earlier to $[3, 11]$ trades accuracy for compute, with the curve decreasing smoothly to -41.3% FLOPs. This smoothness also justifies the per-operator–backbone one-shot sweep of the window and cap in Appendix A.1: this choice is not a sensitive hyperparameter requiring tuning.

(a) Correction threshold β

Variant	Top-1 (%)	GFLOPs	P50	P90
raw max-gap (no corr.)	79.54	3.48	181	181
$\beta = -1.0$	79.68	4.07	103	122
$\beta = -0.75$	79.63	3.77	134	151
$\beta = -0.5$ **(main)**	**79.63**	3.56	160	172
$\beta = -0.3$	79.59	3.54	171	179

(b) Batch size

Batch	Top-1 (%)	GFLOPs	$\mathbb{E}\,\lvert r_x^{[b]} - r_x^{[1]}\rvert$
32	79.63	3.53	14.9
64	79.55	3.57	13.9
128	79.56	3.56	12.9
256	79.57	3.54	12.9

Table 4: Robustness ablations on DeiT-S/16 (ImageNet-1K val). P50 / P90: the median and 90th percentile of the executed compression per image, in tokens removed, characterizing the distribution of the actual compression strength across β values. $\mathbb{E}\,\lvert r_x^{[b]} - r_x^{[1]}\rvert$, where the superscript indexes the batch size b and b=1 recovers the per-image budget: the mean deviation (in tokens) of the batch-shared budget from the per-image budget, measuring the budget shift introduced by within-batch calibration.

Correction threshold. Table 4(a) isolates the tail correction. Removing the correction (raw max-gap) drops the accuracy to 79.54, and its boundary can fall inside high-value plateaus and lead to over-merging, showing that the safety comes from the correction itself, not from the max-gap detection. Accuracy varies by no more than 0.09 percentage points as β ranges over $[-1.0, -0.3]$:

the correction acts directionally (pushing the boundary toward the tail, erring toward keeping more), and the accuracy is insensitive to its exact position. Hence $\beta = -0.5$ is reused as the single constant across all four tasks, without any per-task tuning.

Batch size. TT-ACS shares one budget within the batch, calibrated by a reference image. Table 4(b) shows that the batch-shared budget matches or slightly improves on per-image decisions (79.55–79.63 vs. 79.63), and the budget deviation changes only slightly as the batch grows (14.9→12.9 tokens): the shared budget derived from the hard-to-compress reference image acts as an implicit regularizer against the per-image knee's occasional over-merging on near-flat profiles. The batch size can be set freely and does not affect the operating mechanism of the method.

4.3 Visualization

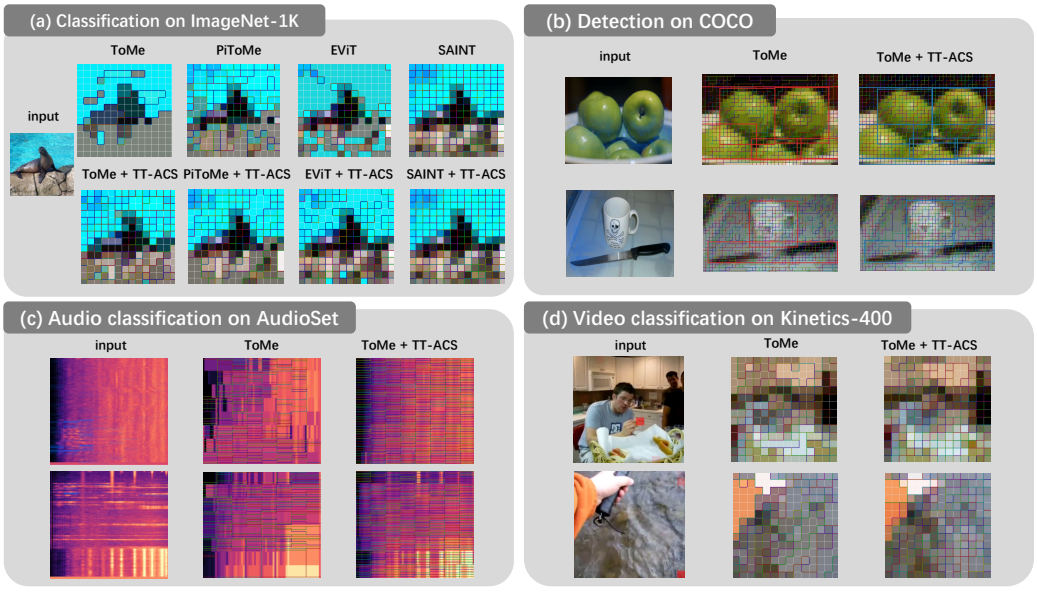

Figure 5: Visual comparison of fixed budgets and TT-ACS adaptive budgets across four tasks.

Figure 5 illustrates what fixed and adaptive budgets actually do to the tokens, and the pattern is the same on all four tasks. The task-aware adaptivity tends to preserve tokens carrying the main subjects as well as rich detail and texture; compared with fixed compression, it merges or prunes more conservatively, so all accuracy metrics improve. Per-operator examples and the layer-by-layer merging process are shown in Appendix A.4.

5 Conclusion and Future Work

TT-ACS revisits a factor of token compression that is usually fixed in advance: the compression budget itself. By separating budget scheduling from the compression operator, TT-ACS estimates task-relative redundancy from task-relevant attention structure and converts it into an adaptive compression budget, while leaving the underlying operator unchanged and requiring no additional training. Across 13 operator–backbone configurations and multiple downstream tasks, the results consistently demonstrate that adaptive budget scheduling provides a complementary dimension to existing token compression operators. At the same time, the current design leaves two practical challenges open: adaptive budget estimation introduces additional decision overhead, while conservative scheduling does not fully exploit the available compression potential. Future work will therefore focus on lower-overhead and finer-grained budget estimation, further advancing token compression from fixed-budget operation toward task-aware dynamic compute allocation.

AI USE STATEMENT

In this work, generative AI tools were used only to polish the written text. We did not use them for the method design, the core scheduling algorithm and its implementation, or the ideation or execution of the research. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies training-free token compression for vision transformers. All experiments use established public benchmarks, namely ImageNet-1K, COCO, Kinetics-400, and AudioSet, under their released evaluation protocols, and involve no human subjects or personal data. The method modifies neither the training data nor the model weights of the frozen backbones, so it introduces no new biases beyond those already present in the underlying pretrained models. By reducing the computation of pretrained models without retraining, TT-ACS lowers the energy and hardware cost of deploying vision transformers, which we consider a positive societal impact. We do not anticipate other immediate ethical concerns.

REPRODUCIBILITY STATEMENT

TT-ACS introduces no training and no learnable parameters; reproducing our results requires only the pretrained checkpoints in Appendices A.1.1–A.1.4 and the decision procedure in the main text and appendix. Section 3 gives the complete formulation (Eqs. (1)–(6)); Appendices A.1.1–A.1.4 provide, for each task, pseudocode (Algorithms 1–4); Appendix A.1 states the measurement protocols (im/s, GFLOPs, and device). Source code covering all four tasks and all operators is available at https://anonymous.4open.science/r/TT-ACS/.

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. *arXiv preprint arXiv:2303.17604*, 2023.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *Int. Conf. Learn. Represent.*, 2023.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Eur. Conf. Comput. Vis.*, 2024.

Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. Diffrate: Differentiable compression rate for efficient vision transformers. *arXiv preprint arXiv:2305.17997*, 2023.

Kyumin Choi and Ikbeom Jang. Rapid: Layer-wise redundancy-aware pruning and importance-driven token merging for efficient vit. *arXiv preprint arXiv:2606.08156*, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2021.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. In *Eur. Conf. Comput. Vis.*, 2022.

Yizeng Han, Guocheng Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11):7436–7456, 2022.

Po-Yao Huang, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer. Masked autoencoders that listen: Learning from speech by denoising. In *Advances in Neural Information Processing Systems*, 2022.

Simin Huo and Ning Li. Mame and mare: Matrix-based token merging and restoration for efficient visual perception and synthesis. *arXiv preprint arXiv:2604.13432*, 2026.

Ahmadreza Jeddi, Negin Baghbanzadeh, Elham Dolatabadi, and Babak Taati. Similarity-aware token pruning: Your vlm but faster. *arXiv preprint arXiv:2503.11549*, 2025.

Jaeyeon Lee and Dong-Wan Choi. Lossless token merging even without fine-tuning in vision transformers. *arXiv preprint arXiv:2505.15160*, 2025.

Semi Lee, Hyejin Go, and Hyesong Choi. Adamerge: Salience-aware adaptive token merging for training-free acceleration of vision transformers. *arXiv preprint arXiv:2605.27465*, 2026.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Evit: Expediting vision transformers via token reorganizations. In *Int. Conf. Learn. Represent.*, 2022.

Henry B. Mann. Nonparametric tests against trend. *Econometrica*, 13(3):245–259, 1945.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

Ewan S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Anthony N. Pettitt. A non-parametric approach to the change-point problem. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(2):126–135, 1979.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems 34*, 2021.

Simon Ravé, Pejman Rasti, and David Rousseau. Mpm: Mutual pair merging for efficient vision transformers. In *CVPR Findings*, 2026.

Byungseok Roh, JaeWoong Shin, Wuhyun Kim, and Sunggoo Jung. Sparse detr: Efficient end-to-end object detection with learnable sparsity. In *Int. Conf. Learn. Represent.*, 2022.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, 2022.

Hoai-Chau Tran, Duy M. H. Nguyen, Duy M. Nguyen, Trung-Tin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Y. Zou, Binh T. Nguyen, and Mathias Niepert. Accelerating transformers with spectrum-preserving token merging. In *Advances in Neural Information Processing Systems 37*, 2024.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2025.

Evelyn Zhang, Bang Xiao, Jiayi Tang, Qianli Ma, Chang Zou, Xuefei Ning, Xuming Hu, and Linfeng Zhang. Token pruning for caching better: 9 times acceleration on stable diffusion for free. *arXiv preprint arXiv:2501.00375*, 2025.

Dehua Zheng, Wenhui Dong, Hailin Hu, Xinghao Chen, and Yunhe Wang. Less is more: Focus attention for efficient detr. In *Int. Conf. Comput. Vis.*, 2023.

APPENDIX

Section A.1 gives the experimental configuration and engineering details, Section A.2 situates TT-ACS against existing adaptive-budget methods (Table 6), Section A.3 validates the task-relevant attention boundary as an estimate of task-relative redundancy (Figure 6), and Section A.4 provides additional token visualizations.

A.1 IMPLEMENTATION DETAILS

Since the backbones and operators differ in structure and operating mechanism, each backbone–operator pair requires sweeping one structural choice (the decision window and $\rho_{\max}$) once; the sweep exchanges accuracy for FLOPs smoothly rather than catastrophically (Table 3(b)), so this choice is not a sensitive hyperparameter. The exact configurations are listed in the Method and TT-ACS columns of Tables 1 and 2. All im/s numbers are the mean $\pm$ std over three CUDA-event repeats in one session; all GFLOPs (MACs) are measured with FlopCounterMode; the experimental device is an RTX 5090 (32GB) with 25 vCPU Intel(R) Xeon(R) Platinum 8470Q, CUDA 12.8, and GPU driver 580.105.08.

The decision granularity also differs by task. Detection decides per image, since the encoder processes each image's own masked sequence of valid tokens. The other three tasks make batch-shared decisions: one budget is decided per batch and applied to all its inputs, so the decision overhead is shared across the batch. In the batch-shared modes, the budget is calibrated by a reference image. Each image receives a scalar redundancy score, defined as the mean minus the median of its sorted importance profile; this score grows with the dominance of the head and hence with the image's compression tolerance. The images of a batch are ranked by this score, and the reference is selected on the hard-to-compress side of the knee of this batch-level redundancy curve, located by the same max-gap rule with tail correction; when the batch is too small for the curve to be informative, the reference falls back to the image at the 20% redundancy quantile, and for a single-image batch, the reference is the image itself. Anchoring the shared budget at the hard-to-compress end keeps it on the conservative side of the per-image budgets and acts as an implicit regularizer against occasional over-merging on near-flat profiles; the deviation from the per-image budget is small and accuracy is preserved.

Table 5 summarizes how each task instantiates the shared decision machinery; the subsections below give each task's pseudocode.

Task	Signal	Decision position	Execution	Deviations from shared machinery
Classification	CLS row	window $[a, b]$	per layer, batch-shared	none
Detection	encoder self-attention (received)	$D \in \{1, 2\}$	per image, single pass	masked valid sources; window after D
Video	received-attention profile	D=2	per layer, batch-shared	`class_token=False`; size-weighted pooling
Audio	CLS row	$[D{+}1, 11]$	per layer, batch-shared	`prop_attn` off; instrumented ρ

Table 5: Task instantiations of the shared decision machinery.

A.1.1 CLASSIFICATION ON IMAGENET-1K

This task is a direct instantiation of the shared machinery: the CLS-row signal and per-layer batch-shared execution inside the window (Algorithm 1).

A.1.2 OBJECT DETECTION ON COCO

The budget signal is available inside the encoder (Section 4.1); the budget is per image and is executed with the official ToMe kernels in a single forward pass, with no profiling pass (Algorithm 2). Exploring the decision position shows the same structure as classification: D=0 yields a flat profile and almost no executed compression, because the shallowest self-attention carries no head–tail structure; D=1 and D=2 give a usable signal; merging in the early layers is harmful, while merging after D with a per-image budget keeps the accuracy loss tolerable.

Algorithm 1 Classification (CLS row, per-layer batch-shared).

Require: frozen ViT; window $[a, b]$; $\rho_{\max}$; $\beta = -0.5$

for each layer $\ell = a, \ldots, b$ **do**
  $A \leftarrow$ pre-softmax CLS-row logits at layer ℓ ▷ $H \times N$; no extra pass
  $p \leftarrow$ head-average of A ▷ Eq. (2), $\mathcal{Q} = \{\text{CLS}\}$
  sort p descending; $z \leftarrow (p - \mu)/\sigma$
  $\hat{k} \leftarrow \arg\max_k \left(z_{(k)} - z_{(k+1)}\right)$
  if $z_{(\hat{k})} > \beta$ **then**
    $k^* \leftarrow \min\{k : z_{(k)} < \beta\}$
  else
    $k^* \leftarrow \hat{k}$
  end if
  if no $z_{(k)} < \beta$ **then** $k^* \leftarrow N$
  end if
  $\rho \leftarrow \min\left((N - k^*)/N,\ \rho_{\max}\right)$ ▷ batch-shared; N is the current token count at layer ℓ
  merge the $\lceil \rho N \rceil$ least-important tokens via ToMe
end for
return CLS prediction

Algorithm 2 Detection (DETR-R50, per image, single pass).

Require: frozen DETR encoder; decision layer $D \in \{1, 2\}$; $\rho_{\max}$; $\beta = -0.5$

run encoder layers $0, \ldots, D$ unmerged
$A \leftarrow$ raw self-attention logits at layer D
$p_i \leftarrow$ average of A over heads, queries, and valid (non-padded) source tokens
knee and tail correction on the z-scores of p ▷ as in Algorithm 1
$\rho_x \leftarrow \min\left((N - k^*)/N,\ \rho_{\max}\right)$ ▷ per image
split the total budget $\rho_x N$ evenly over the layers $\ell > D$, giving per-layer counts r_ℓ
for each encoder layer $\ell > D$ **do**
  merge the r_ℓ least-important tokens via the official ToMe kernels
end for
return decoder output

A.1.3 Video Classification on Kinetics-400

The experiment of Section 4.1 uses the official VideoMAE (Tong et al., 2022) ViT-B checkpoint fine-tuned on Kinetics-400 (top-1 80.9 on the full validation set); the ToMe paper's MaskFeat ViT-L weights are not publicly released. The model has no CLS token (the head is a mean pool over the 1,568 patch tokens followed by a final LayerNorm and a linear classifier), and the integration deviations caused by the missing CLS token are shown in Algorithm 3. Clips are decoded with 16 uniformly sampled frames, bicubic-resized on the shortest edge to 224 and center-cropped (the HF VideoMAE processor path; an antialias-free bilinear resize loses 5.7 pp top-1 and is not used), normalized with the ImageNet statistics; validation uses one center clip per video (1,000 clips in total), at a batch size of 16. Exploring $D \in \{1, \ldots, 5, 7\}$ and $\rho_{\max} \in [0.1, 0.5]$ (36 configurations) shows the same structure as detection: D=1 yields only $\rho \approx 2.8\%$, D=2 gives the clearest head–tail structure with $\rho \approx 24\%$, and $D \geq 3$ is nearly flat; the two deeper-compression rows of Table 2 therefore decide at D=2 and differ only in $\rho_{\max}$, while the lightest tier is decided at D=1, where the knee reads $\rho \approx 2.8\%$ and $\rho_{\max}$ does not bind.

A.1.4 Audio Classification on AudioSet

The experiment of Section 4.1 uses the official Audio MAE (Huang et al., 2022) ViT-B checkpoint fine-tuned on AudioSet-2M (mAP 47.3 on the full eval split; 46.31 on our 17,141-clip subset, which covers all 527 classes). Unlike VideoMAE, this model keeps its CLS token, so the integration follows the classification algorithm with the deviations shown in Algorithm 4. Inputs are 128-mel log-filterbank spectrograms of 1,024 frames with the AudioSet normalization, following the AudioMAE recipe. Sweeping $D \in \{3, \ldots, 11\}$ and $\rho_{\max} \in [0.35, 0.9]$ shows that $\rho_{\max}$ never binds

Algorithm 3 Video (VideoMAE ViT-B, no CLS).

Require: frozen VideoMAE; D=2; $\rho_{\max}$; $\beta = -0.5$; batch-shared
1: decode 16 frames, resize, and normalize; run layers $0, \ldots, D$ unmerged
2: $A \leftarrow$ self-attention logits at layer D
3: $p \leftarrow$ received-attention profile: A averaged over heads and queries
4: knee and tail correction ▷ as in Algorithm 1
5: $\rho \leftarrow \min\big((N - k^*)/N,\ \rho_{\max}\big)$ ▷ batch-shared
6: **for** each layer $\ell = D{+}1, \ldots, 11$ **do**
7: merge via ToMe with `class_token=False`, proportional attention, and size-weighted pooling
8: **end for**
9: **return** mean-pooled patch features $\rightarrow$ linear classifier

(at a fixed window the mAP is identical across all $\rho_{\max}$ values), so the executed budget is set by the knee alone and shrinks monotonically as the window moves later: $\rho \approx 60.4\%$ (D=3), 49.6% (D=4), 39.1% (D=6), 15.7% (D=9), and 0% (D=11). Table 2 reports one decision window per executed-compression tier.

Algorithm 4 Audio (Audio MAE ViT-B).

Require: frozen Audio MAE; window $[D{+}1, 11]$; $\rho_{\max}$; $\beta = -0.5$; batch-shared
1: compute the 128-mel log-filterbank spectrogram (1,024 frames); run layers $0, \ldots, D$
2: **for** each layer $\ell = D{+}1, \ldots, 11$ **do**
3: $A \leftarrow$ pre-softmax CLS-row logits at layer ℓ
4: $p \leftarrow$ head-average of A
5: knee and tail correction; $\rho \leftarrow \min\big((N - k^*)/N,\ \rho_{\max}\big)$ ▷ $\rho_{\max}$ never binds
6: merge via ToMe with `prop_attn` off and size-weighted pooling
7: **end for**
8: **return** CLS prediction

A.2 Comparison with Existing Budget Methods

	DeiT-S/16			DeiT-B/16		
Method	Acc	GFLOPs	im/s	Acc	GFLOPs	im/s
Baseline	79.72	4.60	2158±1	81.74	17.56	959±15
ATS (Fayyaz et al., 2022)	76.82	2.70	524±10	80.94	10.61	500±1
ATM (Lee & Choi, 2025)	79.47	2.27	784±26	80.63	9.01	707±24
DiffRate (Chen et al., 2023)	79.43	2.91	919±1	80.80	10.44	841±9
RAPID (Choi & Jang, 2026)	79.16	2.99	865±18	80.97	11.46	773±2
ToMe+TT-ACS	78.97	2.67±0.05	1010±5	79.38	9.87±0.05	811±2
PiToMe+TT-ACS	78.63	2.88±0.12	500±7	80.46	10.64±0.51	465±10
EViT+TT-ACS	78.95	2.89±0.12	1473±34	80.68	11.22±0.94	1019±9
SAINT+TT-ACS	79.70	4.45±0.03	1296±4	81.70	16.91±0.02	807±3

Table 6: Comparison between TT-ACS and four representative adaptive-budget methods.

We compare TT-ACS with four representative adaptive-budget methods, each embodying a different source of the budget: a fixed per-layer sampling target (ATS), a layer-decayed similarity threshold (ATM), a differentiably searched per-layer prune–merge schedule (DiffRate), and a hand-placed depth split between pruning and merging (RAPID). The code of RAPID is not open-sourced; we implement it ourselves, and our implementation may differ from the official version. The other three are reproduced from official code on the same backbones under an identical protocol. Since AdaMerge is not open-sourced, we regret that it could not be included in the comparison.

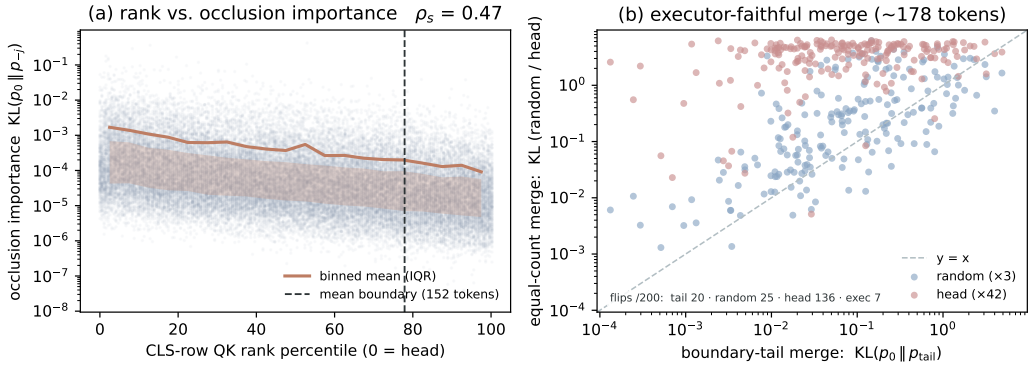

Figure 6: Validating the head–tail boundary derived from task-relevant attention as an estimate of task-relative redundancy on DeiT-S/16. **(a)** Occlusion importance $\mathrm{KL}(p_0 \,\|\, p_{-j})$ of each patch against its rank percentile in the profile, 0 being the highest score; the dashed line marks the mean detected boundary. **(b)** Per-image merging cost under TT-ACS's fixed per-layer counts: the horizontal axis is each image's boundary-tail merge, the vertical axis an equal-count merge of random tokens in blue or highest-scoring tokens in red, and points above the diagonal cost more than merging the tail; the annotation reports incorrect predictions out of 200 per rule.

A.3 Validating the Task-Relevant Attention Boundary as an Estimate of Task-Relative Redundancy

The budget is read from the head–tail boundary of the CLS-row QK profile, so the boundary should mark where task-relevant information ends. Figure 6a checks this on 200 randomly drawn ImageNet-1K val images at the frozen operating point of layers 5–11 with $\rho_{\max}$=0.5 and decisions every two layers, by deleting one patch at a time and measuring how much the prediction changes. If the boundary separates task-relevant tokens from redundant ones, importance should decrease across it. This is what we observe: higher-ranked patches cause larger changes when deleted, with a per-image Spearman ρ of 0.47 ± 0.17, and the averaged importance decreases from head to tail and drops steeply at the detected boundary of 152 tokens on average; beyond the boundary, deleting patches barely changes the prediction, confirming that this region is redundant for the task. Occlusion measures deletion, whereas the executor merges tokens, a milder operation that preserves information through size-weighted averaging. We therefore test the boundary inside the executor itself, keeping TT-ACS's per-layer merge counts fixed and changing only which tokens are merged; in Figure 6b each image's boundary-tail merge serves as the reference, plotted on the horizontal axis against an equal-count merge of random or highest-scoring tokens. If the boundary is a valid budget, merging its tail should rarely cause errors while merging its head should frequently do so. This is also what we observe: spending the counts on the lowest-scoring tokens beyond the boundary leaves 20 of 200 predictions incorrect with a mean KL of 0.33; spending them on random tokens triples the median deviation; spending them on the highest-scoring tokens leaves 136 of 200 incorrect. The same merge counts are therefore safe on the redundant side of the boundary and harmful on the task-relevant side, and the actual selection by bipartite similarity under the same counts performs best, with 7 incorrect predictions, showing that the boundary provides a budget that the unchanged operator can execute safely.

A.4 Additional Token Visualizations

Figure 7 complements Figure 5 with per-operator examples: fixed budgets remove task-relevant foreground tokens, while TT-ACS keeps them. Figure 8 visualizes the layer-by-layer merging process under three decision windows, where the per-layer merge counts shrink as the window moves deeper.



Figure 7: Tokens removed and kept by four fixed-budget operators and by TT-ACS on ImageNet-1K val. Fixed budgets remove task-relevant foreground tokens, whereas TT-ACS preserves them.

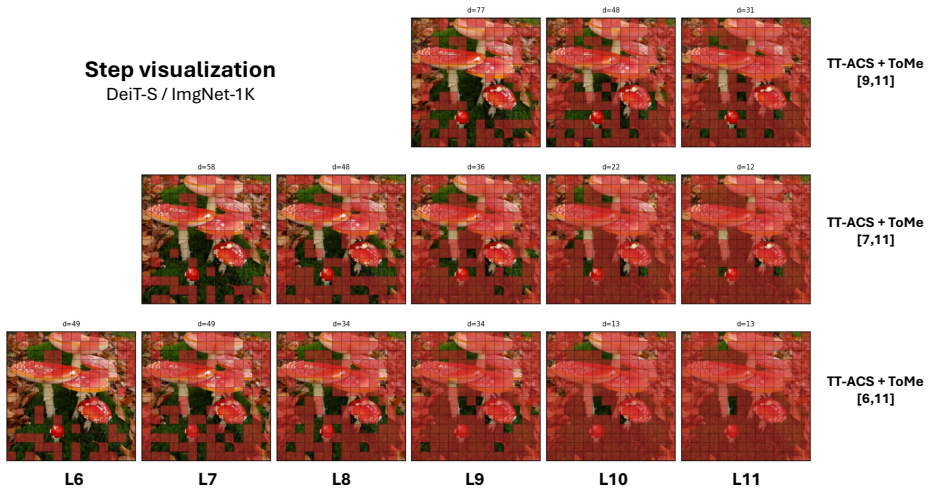

Figure 8: Layer-by-layer merging of TT-ACS+ToMe on DeiT-S/16 (ImageNet-1K) under the $[9, 11]$, $[7, 11]$, and $[6, 11]$ decision windows.