BOTTLENECKED SEMANTIC COMPRESSION BY REPRESENTATION RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision Transformers incur high computational costs when processing long visual token sequences. Existing reduction methods typically rank tokens independently and discard those deemed unimportant or redundant. We propose a Reconstruction-guided framework for visual token condensation(ReConV), which that summarizes the complete visual sequence with a compact Summary Bank (SMB) and retains tokens whose information is not adequately captured by the SMB, as indicated by their reconstruction residuals. At each compression layer, a lightweight position-conditioned decoder reconstructs the current tokens, leaving shared information compressed in the SMB while explicitly preserving poorly reconstructed details. Experiments on image classification and multimodal understanding demonstrate favorable performance–efficiency trade-offs: ReConV surpasses the uncompressed baseline with 48% fewer GMACs on ImageNet and enables reusable image-level representations for efficient multi-turn inference. Code is available at `TokenDrop`.

1 INTRODUCTION

Vision Transformers (ViTs) have become a dominant architecture in modern computer vision, but their self-attention cost grows quadratically with the number of input tokens. This quickly becomes prohibitive for high-resolution images, multiple images, and videos. The problem is even more severe in multimodal large language models (MLLMs), where visual tokens are concatenated with text tokens and jointly processed by an LLM, leading to memory, latency, and throughput burdens.

The redundancy of visual representations has motivated token reduction methods that prune or merge tokens under a constrained computational budget. In conventional visual recognition, token reduction is motivated by the observation that image understanding progressively condenses local patch information into higher-level semantic representations. Accordingly, existing methods estimate token importance Rao et al. (2021); Yin et al. (2022), merge tokens with similar representations Bolya et al. (2023), or formulate reduction as a clustering problem Chang et al. (2023). Most of these methods, however, were developed for single-image classification and are often guided by object-centric discriminative cues. Such assumptions are less suitable for MLLMs, where visual tokens must additionally preserve spatial grounding and information that may become relevant only when combined with a textual instruction.

Consequently, visual token reduction in MLLMs has increasingly relied on text-to-image attention from either the vision encoder or the LLM decoder to rank visual tokens Shang et al. (2024); Chen et al. (2024b); Xing et al. (2024); Chen et al. (2024a; 2026). However, attention primarily reflects local information routing for a particular layer, head, and query, with no guarantee that its magnitude reliably captures token utility or the causal effect of removing a token. It may also be affected by model-induced saliency priors and systematic preferences for particular tokens, and vary across layers and heads. Moreover, accessing attention matrices can interfere with efficient kernels Dao (2024), while decoder-side pruning cannot reduce computation in preceding LLM layers. We summarizes representative visual token reduction paradigms in Table 1. To eliminate dependence on attention, we introduce a learnable Summary Bank (SMB) that semantically aggregates the complete visual sequence, and a lightweight position-conditioned decoder that reconstructs the original tokens from this compact representation. As illustrated in Figure 1, this reconstruction-based criterion requires neither attention distributions nor textual prompts and enables visual token reduction before the LLM.

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.

Table 1: Comparison of representative visual token reduction paradigms. A ✓ denotes explicit design support, whereas an ✗ denotes no explicit guarantee. *No attention-layer/head dependence* excludes reliance on specific attention layers or heads; *prompt independence* excludes textual conditioning; *reduction before the LLM* means compression precedes LLM processing; *preserving uncaptured information* retains information missing from the compressed representation; and *joint token information coverage* evaluates tokens relative to the selected set rather than independently.

Method	No attention-layer / head dependence	Prompt independence	Reduction before the LLM	Preserves uncaptured information	Joint token redundancy / coverage
Decoder-attention ranking	✗	✗	✗	✗	✗
Encoder-attention ranking	✗	✓	✓	✗	✗
Other learned scoring / gating	✓	✓	✓	✗	✗
Similarity-based clustering / merging	✓	✓	✓	✗	✓
Summary Bank + residual(Ours)	✓	✓	✓	✓	✓

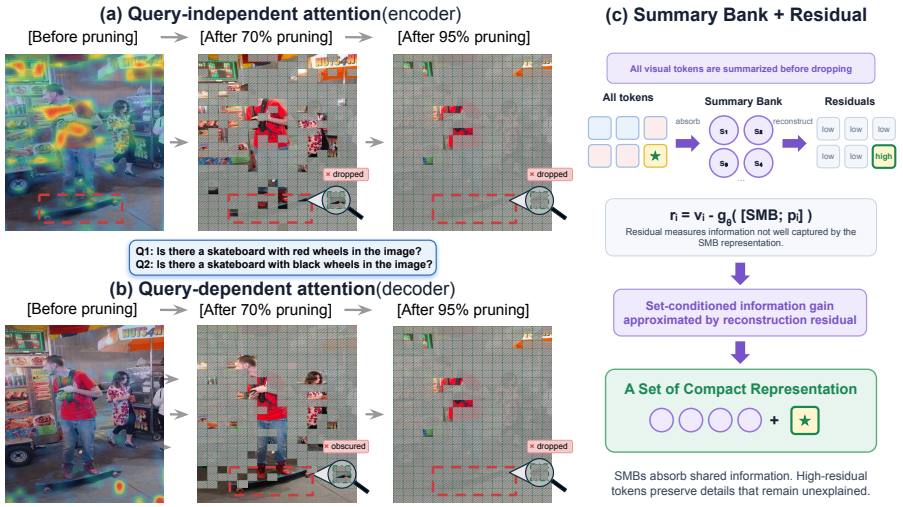

Figure 1: Illustration of visual token pruning. Prompt-independent saliency emphasizes irrelevant regions, while prompt-conditioned pruning can still discard task-relevant details, especially at high pruning ratios. ReConV instead preserves information through SMB summaries and residual tokens.

A second limitation is that existing scoring methods evaluate tokens independently, potentially retaining redundant high-ranking tokens while discarding subtle evidence. Although merging Shang et al. (2024) accounts for redundancy, it may obscure details absent from the compressed representation. ReConV instead encodes the complete visual sequence into the SMBs and uses reconstruction residuals to identify uncaptured information: well-reconstructed tokens are removed, while high-residual tokens are retained as complementary evidence. Selection therefore depends on what the SMB has jointly captured, bridging pruning and information aggregation. Our method retains 100.1% of the baseline classification performance with 48% fewer GMACs and 89% of multi-turn performance using only 5% of the visual tokens. Patch-to-SMB assignments also reveal emergent semantically coherent aggregation (Figure 3).

Our main contributions are as follows:

1. We propose ReConV, a reconstruction-guided framework that compresses visual tokens through a compact Summary Bank (SMB) and lightweight position-conditioned reconstruction.
2. We propose a set of learnable Summary Bank (SMB) tokens that aggregate the complete visual sequence through asymmetric cross-attention. Reconstruction residuals measure information not captured by this joint summary and guide token retention. Patch-to-SMB assignments further reveal emergent semantic grouping.
3. ReConV generalizes across classification and multimodal dialogue, matching the uncompressed classification baseline with 48% fewer GMACs and enabling efficient multi-turn inference through reusable image-level representations.

2 Related Work

Token reduction in ViTs is motivated by the observation that accurate recognition often requires only a subset of the visual tokens. Early methods such as DynamicViT Rao et al. (2021), EViT Liang et al. (2022), Evo-ViT Xu et al. (2022), A-ViT Yin et al. (2022), and ATS Fayyaz et al. (2022) estimate token importance using attention signals or learned scoring functions and remove low-scoring tokens during inference. To preserve information that would otherwise be discarded, subsequent methods such as PPT Wu et al. (2023), ToMe Bolya et al. (2023), Token Pooling Marin et al. (2023), LookupViT Koner et al. (2024), and STViT Chang et al. (2023) compress tokens through merging, pooling, or clustered latent representations. Our method instead uses reconstruction-guided Summary Bank (SMB) tokens as a latent bottleneck, avoiding direct importance scoring and irreversible merging.

Visual token reduction in MLLMs is often text-guided. SparseVLM Zhang et al. (2024) uses decoder cross-attention and recycles pruned tokens through density-peak aggregation, while FastV Chen et al. (2024b) and PDrop Xing et al. (2024) prune according to LLM or instruction-to-image attention. PruMerge Shang et al. (2025) and VisionZip Yang et al. (2025) employ merging, whereas AdaptVision Lin et al. (2025) selectively acquires high-resolution crops. In contrast, our vision-only, reconstruction-guided framework requires neither prompts, attention heuristics, nor region-grounding supervision, enabling reusable image-level compression across ViTs and MLLMs.

Despite their differences, these methods rely on two approximations: *attention magnitude is a reliable proxy for token utility*, and *independent top-K scoring approximates subset optimization*, although a token's utility depends on what has already been preserved. Consequently, redundant high-scoring tokens may be retained while complementary evidence is discarded (Fig. 1). We address both limitations with an SMB that summarizes the complete sequence and reconstruction residuals that measure each token's uncaptured information relative to this summary. Sec. 3 details the method.

3 Methodology

Given visual token embeddings $\mathbf{V}^{(\ell)} \in \mathbb{R}^{B\times N_\ell \times D}$ from a standard vision transformer layer, where N denotes the original token length before compression, our compressive layers learn a small set of *summary bank* (SMB) tokens that summarize the original token set, and discard only those tokens that are already well explained by this learned summary while forwarding the retained tokens to the next layer. Each compressive layer takes the current token set, the current SMB, and positional embeddings as input, produces an updated SMB and a reduced token set, reconstructs tokens with a two-layer MLP decoder, and retains only the top-k tokens according to a residual-based criterion, where k is determined by a prescribed keep ratio. The retained token embeddings are then forwarded to the next layer. Compressive layers can be inserted progressively between transformer encoder blocks, trained from scratch with normal encoder layers or applied as a separate module after a pretrained and frozen vision encoder, as in our MLLM setting. We next describe a single compressive layer in detail, shown in Figure 2. The pseudo-code for a compressive layer is provided in Appendix A.1.

3.1 Summary Bank

Let $\mathbf{V}^{(\ell-1)} \in \mathbb{R}^{B\times N_{\ell-1}\times D}$ denote the visual token sequence entering the ℓ-th compressive layer, and let $\mathbf{S}^{(\ell-1)} \in \mathbb{R}^{B\times M\times D}$ denote the Summary Bank (SMB) propagated from the preceding compressive layer, where $N_0 = N$, $1 \le N_{\ell-1} \le N$ and $M \ll N$. The SMB is initialized from a learnable parameter $\mathbf{S}_0 \in \mathbb{R}^{1\times M\times D}$, broadcast along the batch dimension. Each compressive layer updates the SMB through asymmetric cross-attention, using the SMB tokens as queries and the visual tokens as keys and values:

$$\widetilde{\mathbf{S}}^{(\ell)} = \mathbf{S}^{(\ell-1)} + \mathrm{MHA}\Big(\mathrm{LN}(\mathbf{S}^{(\ell-1)}), \mathrm{LN}(\mathbf{V}^{(\ell-1)}), \mathrm{LN}(\mathbf{V}^{(\ell-1)})\Big). \tag{1}$$

After cross-attention, we further refine $\widetilde{\mathbf{S}}^{(\ell)}$ with an MLP, optionally followed by an SMB-side multi-head self-attention block. When a class token is present, it participates only in these refinement operations and is excluded from SMB-to-visual cross-attention and token reconstruction. We denote the resulting summary bank by $\mathbf{S}^{(\ell)}$. In the following subsections, we omit the layer index and use $\mathbf{S}$ and $\mathbf{s}_m$ to denote the updated summary bank and its m-th token, respectively.

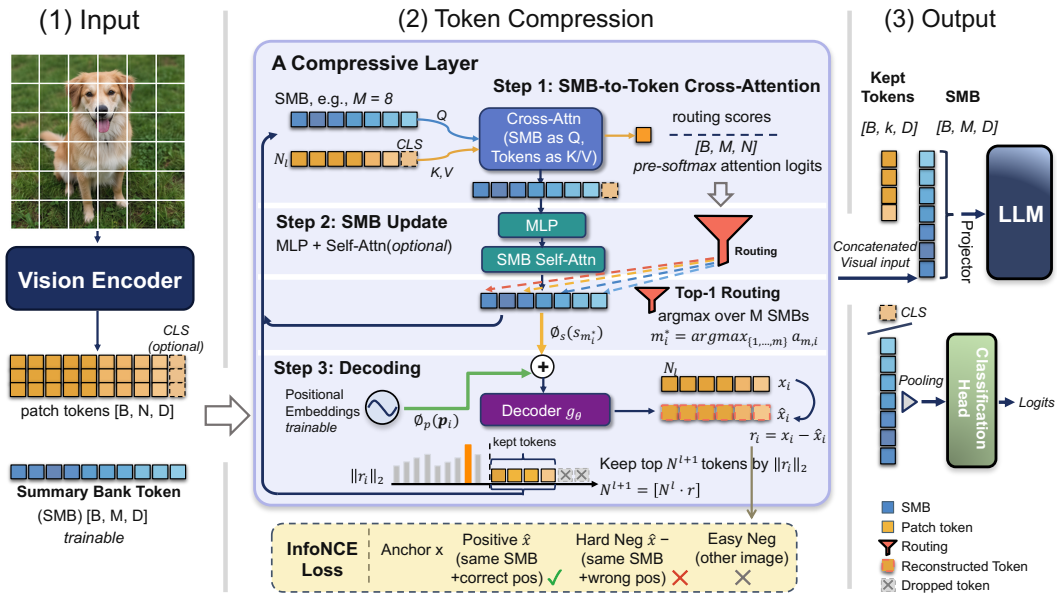

Figure 2: Overview of ReConV, the proposed reconstruction-guided token pruning framework. Given N token embeddings from a vision encoder or transformer layer, each compressive layer uses a small set of learned summary bank (SMB) tokens to summarize the input via cross-attention, reconstructs the original tokens using the SMB representation and positional information, and retains the top-k tokens with the largest reconstruction residuals (approximately $N \cdot r$, where r is the keep ratio). Tokens that are well reconstructed are dropped as redundant, while the retained tokens are passed to the next layer. For classification, the model uses the CLS token when available, or pooled SMB features otherwise; for other tasks, it uses the retained visual tokens with SMBs.

3.2 Token Routing and Reconstruction

The pre-softmax attention logits from SMB-to-token cross-attention, averaged across heads, are reused as routing scores. For each visual token $\mathbf{v}_i$, we select the summary slot with the highest score:

$$m_i^\star = \arg \max_{m \in \{1,\dots,M\}} a_{m,i}, \tag{2}$$

where $a_{m,i}$ denotes the attention logit between the m-th SMB token $\mathbf{s}_m$ and visual token $\mathbf{v}_i$. This top-1 routing assigns each visual token to a single SMB token, requiring only one decoder evaluation per token. To preserve token-level spatial identity, we maintain a learnable positional table $\mathbf{Pos} \in \mathbb{R}^{1\times N\times D}$. The original position indices of retained tokens are propagated across compressive layers, allowing the corresponding positional embedding $\mathbf{p}_i$ to be retrieved at each layer. Given the routed SMB token $\mathbf{s}_{m_i^\star}$ and positional embedding $\mathbf{p}_i$, we project both into the decoder space, concatenate them, and reconstruct the input token using a decoder g_θ:

$$\hat{\mathbf{v}}_i = g_\theta\big([\phi_s(\mathbf{s}_{m_i^\star}); \phi_p(\mathbf{p}_i)]\big)\,. \tag{3}$$

The reconstructed token $\hat{\mathbf{v}}_i$ is used to compute the reconstruction residual for token selection and to define the training objective described in Section 3.3. Although no semantic supervision is imposed on the routing assignments, visualizing them reveals the spontaneous emergence of semantically coherent patch groups, as shown in Figure 3. Additional examples are provided in Appendices B.3 and B.1.1.

3.3 Reconstruction Objective and Token Selection

We optimize the compression module with an InfoNCE reconstruction objective, while token selection is guided by the orthogonal residual derived from each reconstructed token. Given the reconstructed

token $\hat{\mathbf{v}}_i$, we compute its normalized direction $\mathbf{u}_i$ and define the orthogonal residual for token selection as:

$$\mathbf{u}_i = \frac{\hat{\mathbf{v}}_i}{\|\hat{\mathbf{v}}_i\|_2}, \qquad \mathbf{r}_i = \mathbf{v}_i - (\mathbf{v}_i^\top \mathbf{u}_i)\mathbf{u}_i. \tag{4}$$

The token-selection score is given by $|\mathbf{r}_i|_2$. The top-k tokens with the largest residual norms are retained, while the remaining tokens are dropped. Intuitively, this preserves tokens that remain poorly explained after removing the component aligned with their reconstructions. The reconstruction is optimized with a contrastive objective. Specifically, we normalize the original and reconstructed tokens and define the positive and negative contrastive logits as

$$\bar{\mathbf{v}}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}, \quad \bar{\hat{\mathbf{v}}}_j = \frac{\hat{\mathbf{v}}_j}{\|\hat{\mathbf{v}}_j\|_2}, \qquad \ell_i^+ = \frac{\bar{\mathbf{v}}_i^\top \bar{\hat{\mathbf{v}}}_i}{\tau}, \quad \ell_{i,j}^- = \frac{\bar{\mathbf{v}}_i^\top \bar{\hat{\mathbf{v}}}_j}{\tau}, \quad j \in \mathcal{N}_i, \tag{5}$$

where τ is the contrastive temperature and $\mathcal{N}_i$ denotes the negative set associated with anchor i. Anchors are sampled from a candidate pool Ω, defined as either the dropped-token set or the full token set. We sample an anchor set $\mathcal{T} \subseteq \Omega$ uniformly without replacement, with

$$|\mathcal{T}| = \begin{cases} |\Omega|, & T_0 = 0, \\ \min(T_0, |\Omega|), & T_0 > 0, \end{cases} \tag{6}$$

where $T_0 \geq 0$ is a pre-specified anchor budget, with $T_0 = 0$ indicating that all candidates in Ω are used. For each anchor i, hard negatives are reconstructed tokens at different positions within the same image that are routed to the same SMB slot, whereas easy negatives are reconstructed tokens from other images in the batch. The resulting reconstruction loss is

$$\mathcal{L}_{\text{rec}}^{\text{InfoNCE}} = -\frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \log \frac{\exp(\ell_i^+)}{\exp(\ell_i^+) + \sum_{j \in \mathcal{N}_i} \exp(\ell_{i,j}^-)}. \tag{7}$$

Thus, the same reconstruction serves both token selection and module optimization: the orthogonal residual determines which tokens are retained, while the InfoNCE objective trains the compression module to recover token-specific representations. At compressive layer ℓ, we retain $k_\ell = \text{round}(N_{\ell-1}\rho_\ell)$ tokens with the largest residual norms, where ρ_ℓ is the prescribed keep ratio. The retained representations forwarded to the next layer can be either the original token embeddings $\mathbf{v}_i$ or their residuals $\mathbf{r}_i$. Forwarding the residuals explicitly propagates complementary information not captured by the SMB reconstruction. Across L compressive layers, the number of retained non-SMB tokens is approximately

$$N_{\text{ret}} \approx N \prod_{\ell=1}^{L} \rho_\ell. \tag{8}$$

When the SMB is included in the final compressed representation, its total length is approximately $M + N_{\text{ret}}$. Finally, the model is trained by combining the task and reconstruction objectives:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{rec}}^{\text{InfoNCE}}, \tag{9}$$

where λ controls the contribution of the reconstruction objective.

4 Experiments

4.1 Image Classification

4.1.1 Experimental Setup

We evaluate image classification on ImageNet-1K Deng et al. (2009) using 224×224 inputs. Following prior work, we adopt DeiT-S Touvron et al. (2021) as our primary backbone and insert compressive layers at designated positions within it. For example, A1A3A5 denotes inserting a compressive layer after each of Transformer blocks 1, 3, and 5. We set the reconstruction loss weight



Figure 3: Patch-to-SMB assignments for tokens entering the first, second, and third compression layers of our A1A3A5 classification model, which use keep ratios of 0.75, 0.25, and 0.0, respectively. The model's performance is reported in Table 2. Green numbers identify the SMB tokens assigned to reconstruct the corresponding patches, and patches assigned to the same SMB are masked with the same color. Unnumbered patches have been dropped by preceding layers. Examples are randomly selected from the ImageNet validation set.

to $\lambda = 0.05$ for all experiments. When distillation is enabled, we apply a soft KL-divergence loss with a DeiT-S model trained from scratch as the teacher. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda\, \mathcal{L}_{\text{rec}}, \quad \mathcal{L}_{\text{task}} = (1 - \lambda_{\text{kd}})\, \mathcal{L}_{\text{ce}} + \lambda_{\text{kd}}\, \mathcal{L}_{\text{distill}}, \tag{10}$$

where we set $\lambda_{\text{kd}} = 0.5$ and use a temperature of 1.0 for the distillation loss. We optimize all models with AdamW using a learning rate of 10^{-3}, weight decay of 0.05, and $\beta = (0.9, 0.999)$, together with cosine learning-rate decay to 10^{-5}. Training uses a batch size of 1024, and validation uses a batch size of 2048 on a single H200 GPU with 141GB memory. We report Top-1 accuracy.

4.1.2 Main Results

Table 2 shows that our method achieves a stronger efficiency–accuracy trade-off than prior approaches. We obtain 80.4% Top-1 accuracy while reducing GMACs by 38% relative to DeiT-S; even under a more aggressive 48% reduction, our model still matches the backbone accuracy. We outperform TokenPooling, which achieves 100.2% accuracy but reduces FLOPs by only 3.7% relative to DeiT-S. In addition, we surpass methods that rely on importance-based pruning, such as Dynamic ViT Rao et al. (2021), A-ViT Yin et al. (2022), and ATS Fayyaz et al. (2022), suggesting that hard token dropping based solely on saliency can be destructive when the removed tokens might need to support deeper semantic abstraction. Meanwhile, we achieve a better accuracy trade-off than EViT Liang et al. (2022), Evo-ViT Xu et al. (2022), PPT Wu et al. (2023), and ToMe Bolya et al. (2023), which preserve information through token merging or pooling. These results indicate that SMB can more effectively absorb visual information into a compact global representation.

SMB Utilization and Semantic Partitioning As shown in Fig. 3, patch-to-SMB assignments form spatially contiguous, semantically related regions despite receiving no spatial or semantic supervision. This suggests that the SMBs do not merely aggregate patches according to low-level proximity, but naturally develop specialization for recurring visual structures. Figure 3 also shows that the residual-norm dropping policy preferentially removes tokens from slowly varying regions, while retaining tokens near object boundaries and other high-frequency details. This is consistent with our design, which removes redundant structure while preserving visually informative content. We provide quantitative ablations in Appendix B.1.1. Comparing SMB utilization maps from single- and three-layer compression models reveals more pronounced aggregation in the latter; similar behavior also emerges in multimodal tasks(see Appendix B.3).

Table 2: Comparison of computational cost and accuracy on ImageNet. The percentage in GMACs indicates the reduction computation relative to the backbone, and the percentage in accuracy indicates the retained performance relative to the backbone accuracy. The configurations "0.75,0.25,0" denote the keep ratios of the three compressive layers, corresponding to ρ_1, ρ_2, ρ_3 in Section 3.3. Our$_{distill}$ indicates that a DeiT-S teacher trained from scratch is used for distillation.

Model	GMACs	Accuracy
DeiT-S(Base)	4.61 (100%)	79.8 (100%)
A-ViT Yin et al. (2022)(CVPR'22)	3.6 ($\downarrow$ 22%)	78.6 (98.5%)
STViT Chang et al. (2023)(CVPR'23)	3.16 ($\downarrow$ 31%)	80.6 (101.0%)
EViT Liang et al. (2022)(ICLR'22)	3.0 ($\downarrow$ 35%)	79.5 (99.6%)
Evo-ViT Xu et al. (2022)(AAAI'22)	3.0 ($\downarrow$ 35%)	79.4 (99.5%)
Dynamic ViT Rao et al. (2021)(NeurIPS'21)	2.9 ($\downarrow$ 37%)	79.3 (99.4%)
ATS Fayyaz et al. (2022)(ECCV'22)	2.9 ($\downarrow$ 37%)	79.7 (99.9%)
PPT Wu et al. (2023)(CoRR'22)	2.9 ($\downarrow$ 37%)	79.8 (100.0%)
ToMe Bolya et al. (2023)(ICLR'23)	2.7 ($\downarrow$ 37%)	79.4 (99.5%)
ReConV(Ours$_{\text{distill}}$-A1A3A5[0.75,0.25,0.25])	2.88 ($\downarrow$ 38%)	80.40 (100.8%)
ReConV(Ours$_{\text{distill}}$-A1A3A5[0.5,0.14,0.0])	2.42 ($\downarrow$ 48%)	79.90 (100.1%)

We analyze the SMB usage distribution over the full validation set and find that the SMBs are used approximately uniformly. For the A1A3A5 model, the corresponding entropies for the three compressive layers are 2.7030, 2.7531, and 2.7418 nats. These results indicate that the model makes effective use of the available SMB budget. For each SMB, we aggregate all tokens assigned to it across the validation set and construct a class distribution using the labels of their source images. For the 16-SMB model, the entropy values range from 6.1162 to 6.8889 nats, with a mean of 6.8219, close to the maximum of 6.9078 nats for 1000 classes. This high entropy indicates that no SMB is dominated by a small subset of classes. Together with the coherent assignment maps, this suggests that SMBs capture class-agnostic mid-level visual structure rather than acting as class-specific experts.

4.2 Multi-Turn Multimodal Understanding

4.2.1 Experimental Setup

We instantiate ReConV on LLaVA-1.5 Liu et al. (2024a), using CLIP-ViT-L-336px as the frozen vision encoder and Vicuna-13B as the LLM backbone, connected by an MLP projector. Two lightweight compressive layers are inserted between the vision encoder and projector, and the compressor and projector are trained from scratch. During pretraining, we optimize $\mathcal{L} = \mathcal{L}\text{task} + \lambda\mathcal{L}\text{rec}$ with $\lambda = 0.01$. During instruction tuning, the projector is frozen and the LLM is updated with LoRA, while λ is reduced to 0.001 to preserve the learned compact representation during task adaptation. We use the 558K image-text alignment dataset and 665K multimodal instruction dataset released with LLaVA-1.5 for the two stages, respectively. Details are provided in Appendix B.2. All training runs use $4\times$ H200 GPUs with 141 GB each, and evaluation uses $4\times$ L40S GPUs with 48 GB each.

4.2.2 Main Results

Evaluation protocol. Multi-turn dialogue is a natural setting for image-level compression because multiple questions share the same image. ReConV compresses each image once and reuses its visual representation across turns, amortizing visual encoding and compression costs. In contrast, prompt-conditioned selection depends on the current query and must be recomputed whenever the prompt changes. We evaluate this capability on MT-VQA-v2 Goyal et al. (2017) and ConvBench Liu et al. (2024b). MT-VQA-v2 contains 3–14 questions per image, which we treat as successive independent queries, whereas ConvBench comprises native conversations with cross-turn dependencies. MetaCompress Wang et al. (2026) evaluates a similar setting using a manually constructed VQAv2 subset with fixed three-turn dialogues; however, neither the subset nor its implementation is publicly available, precluding a direct comparison. For ConvBench, we report direct-grading scores from Qwen3-VL-8B-Instruct on a 0–10 scale, using the same judge and prompt for all methods.

At a 95% token reduction rate, ReConV achieves the best MT-VQA-v2 overall accuracy and ConvBench average among compressed methods, while providing $2.73\times$ and $1.51\times$ speedups, respec-



Question ID	Question	LLaVA 1.5	Ours
380531000	Are the trees blurry?	yes	yes
380531001	Is there a forest in the picture?	yes	no
380531002	Is there a mountain?	no	no
380531003	Can passengers fly to Europe in this?	yes	yes
380531004	Is this plane taking off or landing?	taking off	taking off
380531005	What is flying in the sky?	airplane	airplane
380531006	Is the plane getting ready to land?	yes	no
380531007	What direction are the planes pointed?	left	left
380531008	What is beside the blue water?	rocks	rocks
380531009	Does the body of water look calm?	yes	yes



Question ID	Question	LLaVA-1.5	Ours
127200000	What are they eating?	pizza	pizza
127200001	How old is this baby?	2	2
127200002	Are the kids eating birthday cake?	no	no
127200003	Are the children smiling?	yes	yes
127200004	How many animals are on the cake?	0	0

Figure 4: Examples from the MT-VQAv2 test split, where the number of questions associated with each image varies across samples. In the multi-turn setting, each image is encoded and optionally compressed only once, and the resulting visual embeddings are reused across all associated questions.

Table 3: Comparison on MT-VQA-v2 and ConvBench at a 95% visual token reduction rate. *Amort.* denotes latency per question with image encoding amortized across questions, while *Mean Total Prefill* denotes total prefill latency averaged across conversations.

Method	MT-VQA-v2						ConvBench					
	Yes/No	Number	Other	Overall	Amort. (ms/q) ↓	Speedup (×) ↑	S_1	S_2	S_3	Average	Mean Total Prefill (s) ↓	Speedup (×) ↑
Uncompressed	93.28	62.22	72.22	79.99	314.50	1.00	3.47	4.33	4.68	4.16	361.05	1.00
Random Pruning	87.19	47.43	59.97	69.76	114.91	2.74	3.29	3.29	2.60	3.06	242.06	1.49
Spatial Pooling	80.41	38.60	50.58	61.51	114.32	2.75	3.12	3.12	2.77	3.00	240.92	1.50
FastV[3]	80.56	40.75	54.15	63.51	192.40	1.63	3.81	3.64	3.47	3.64	739.47	0.49
PDrop[28]	75.71	30.56	42.05	54.61	119.30	2.64	2.60	2.95	2.60	2.72	237.76	1.52
PruMerge+[22]	83.38	45.67	59.94	67.97	117.17	2.68	3.72	3.80	3.46	3.66	239.79	1.51
SparseVLM+[32]	79.55	32.78	48.89	59.69	126.79	2.48	2.08	3.99	1.91	2.66	238.65	1.51
ReConV(Ours)	86.85	50.82	61.92	70.93	115.36	2.73	3.12	3.81	4.16	3.70	239.81	1.51

tively. This favorable accuracy–efficiency trade-off under extreme compression validates the SMB's effectiveness in absorbing information from the complete visual sequence.

4.3 Ablation Study

We analyze the accuracy–compute trade-off by varying compressive-layer placement, keep ratios, and the number of SMBs. Figure 4.3 (Left) shows that, for both 8 and 16 SMBs, earlier compression reduces computation but degrades accuracy, indicating that aggressive reduction is more harmful in shallow layers. The two-layer A3A5 model with keep ratios of 0.19 and 0.0 lies on the same approximately linear frontier as the A3 and A5 configurations, while reducing the SMB count from 16 to 8 produces a less favorable trade-off. Additional results are reported in Appendix B.1.1.

Figure 4.3 (Right) further examines different keep-ratio schedules for models with 8, 16, and 32 SMBs, ranging from conservative schedules with less reduction in early layers to aggressive schedules with stronger early-layer compression. The annotated values indicate the keep ratios applied at each compressive layer. Across these configurations, the models exhibit a consistent and approximately linear accuracy–compute trend, suggesting that the proposed framework is robust to different configuration choices. Notably, at a similar computational cost, the 32-SMB model with a $0.75, 0.25, 0.0$ keep-ratio schedule outperforms the 8-SMB model with a $0.80, 0.50, 0.50$ schedule, indicating that a larger summary bank provides a more effective compression basis for classification.

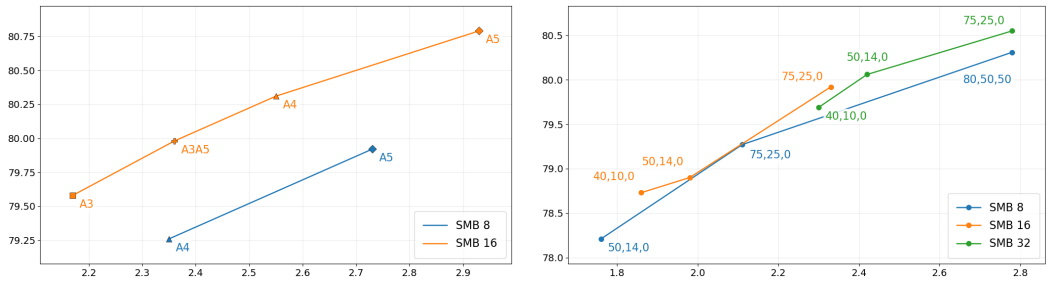

Figure 5: ImageNet1K top-1 accuracy percentage vs computation cost (GMACs) tradeoff. Left: Models with 1-2 compressive layers with varying number of SMBs and layer placement. Right: Models with 3 compressive layers (A1A3A5) with varying number of SMBs and keep ratios.

Table 4: Ablation study of the compression design on multimodal understanding. *Inter-layer repr.* and *LLM input* specify whether original or residual tokens are propagated between compression layers and supplied to the LLM, respectively. K denotes the total token budget, including retained and SMB tokens. *MME-P* and *MME-C* denote the total perception and cognition scores.

Ablation	Inter-layer repr.	LLM input	Position reinjection	Total tokens	# Retained tokens	# SMB tokens	# Compression layers	MME-P Score	MME-C Score
LLaVA-1.5-13B (Base)			*Uncompressed (576 tokens)*					1531	295
Representation propagation	Original	Original	✓	127	119	8	2	1242	273
	Original	Residual	✓	127	119	8	2	1370	285
	Residual	Original	✓	127	119	8	2	1405	305
	Residual	Residual	✓	127	119	8	2	1457	291
Positional information	Residual	Residual	✗	127	119	8	2	1337	285
	Residual	Residual	✓	127	119	8	2	1457	291
Allocation ($K = 32$)	Residual	Residual	✓	32	1	31	2	1337	291
	Residual	Residual	✓	32	16	16	2	1304	293
	Residual	Residual	✓	32	24	8	2	1352	288
Allocation ($K = 64$)	Residual	Residual	✓	64	32	32	2	1327	321
	Residual	Residual	✓	64	56	8	2	1404	321
Token budget (8 SMBs)	Residual	Residual	✓	127	119	8	2	1457	291
	Residual	Residual	✓	144	136	8	2	1423	286
	Residual	Residual	✓	192	184	8	2	1457	282
	Residual	Residual	✓	263	255	8	2	1519	289
Compression depth	Residual	Residual	✓	32	24	8	2	1352	288
	Residual	Residual	✓	32	24	8	3	1325	312

On MME Fu et al. (2023), we ablate compression depth, positional reinjection, token allocation, and representation type. Re-adding positional embeddings before each compression layer improves MME-P from 1337 to 1457 and MME-C from 285 to 291. Under a fixed budget, allocating more tokens to explicit retention generally performs better: at $K = 64$, changing the retained/SMB allocation from $32 + 32$ to $56 + 8$ raises MME-P from 1327 to 1404 while preserving MME-C. Using residuals for both inter-layer propagation and LLM input achieves the best overall performance, supporting their role in propagating information not absorbed by the SMB. Across budgets, retaining approximately 50% of the visual tokens reduces MME-P by only 0.8%, while a 95% reduction incurs an 11.7% drop, demonstrating a favorable performance–efficiency trade-off.

5 Conclusion

In this paper, we introduced a reconstruction-guided framework for visual token compression. By compressing token sequences into a compact set of learnable Summary Bank tokens and using reconstruction residuals to guide token retention, our method enables hard token dropping without hard semantic deletion. Experiments on image classification and multimodal understanding demonstrate favorable performance–efficiency trade-offs across tasks.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations (ICLR)*. OpenReview.net, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Shuning Chang, Pichao Wang, Ming Lin, Fan Wang, David Junhao Zhang, Rong Jin, and Mike Zheng Shou. Making vision transformers efficient from a token sparsification view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6195–6205, 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 *European Conference on Computer Vision (ECCV)*, pp. 19–35. Springer, 2024a.

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 *European Conference on Computer Vision*, pp. 19–35. Springer, 2024b.

Zengjie Chen, Yuxiang Cai, Jingcai Guo, Taotao Cai, Jianwei Yin, and Zhi Chen. Accelerating multimodal large language models with prior-corrected token reduction. *arXiv preprint arXiv:2606.24156*, 2026.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In *Computer Vision – ECCV 2022*, pp. 396–414. Springer, 2022.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Rajat Koner, Gagan Jain, Prateek Jain, Volker Tresp, and Sujoy Paul. Lookupvit: Compressing visual information to a limited number of tokens. In *Computer Vision – ECCV 2024*, pp. 322–337. Springer, 2024. doi: 10.1007/978-3-031-73016-0_19.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 292–305, 2023.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Evit: Expediting vision transformers via token reorganizations. In *The Tenth International Conference on Learning Representations (ICLR)*. OpenReview.net, 2022. URL https://openreview.net/forum?id=BjyvwnXXVn_.

Zichuan Lin, Yicheng Liu, Yang Yang, Lvfang Tao, and Deheng Ye. Adaptvision: Efficient vision-language models via adaptive visual acquisition. *arXiv preprint arXiv:2512.03794*, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 26296–26306, 2024a.

Shuo Liu, Kaining Ying, Hao Zhang, Yue Yang, Yuqi Lin, Tianle Zhang, Chuanhao Li, Yu Qiao, Ping Luo, Wenqi Shao, et al. Convbench: A multi-turn conversation evaluation benchmark with hierarchical capability for large vision-language models. *arXiv preprint arXiv:2403.20194*, 2024b.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024c.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022.

Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers for image classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 12–21, 2023.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *CoRR*, abs/2106.02034, 2021. URL https://arxiv.org/abs/2106.02034.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 12786–12797, 2021.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22857–22867, 2025.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8317–8326, 2019.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Yi Wang, Haofei Zhang, Qihan Huang, Anda Cao, Gongfan Fang, Wei Wang, Xuan Jin, Jie Song, Mingli Song, and Xinchao Wang. Rethinking token reduction for large vision-language models. *arXiv preprint arXiv:2603.21701*, 2026.

Xinjian Wu, Fanhu Zeng, Xiudong Wang, and Xinghao Chen. Ppt: Token pruning and pooling for efficient vision transformers. *CoRR*, abs/2310.01812, 2023. URL https://arxiv.org/abs/2310.01812.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2964–2972, 2022.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19792–19802, 2025.

Hongxu Yin, Arash Vahdat, José M. Álvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10809–10818, 2022.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

AI USE STATEMENT

Generative AI tools were used solely to improve the grammar, clarity, and readability of author-written text. They were not used to develop the research methodology, conduct experiments, analyze results, or formulate scientific claims. All AI-assisted edits were reviewed and revised by the authors, who take full responsibility for the final content of this paper.

Algorithm 1 A Reconstruction-Guided Compressive Layer

Require: tokens $\mathbf{X} \in \mathbb{R}^{N\times D}$, summary bank $\mathbf{S} \in \mathbb{R}^{M\times D}$, positions $\mathbf{P}$, keep ratio ρ, loss type $\tau \in \{\texttt{L2}, \texttt{NCE}\}$
Ensure: updated bank $\mathbf{S}'$, kept tokens $\mathbf{X}_{\text{keep}}$, loss $\mathcal{L}_{\text{rec}}$

$(\Delta\mathbf{S}, \mathbf{L}) \leftarrow \text{CrossAttnWithLogits}(\text{LN}(\mathbf{S}), \text{LN}(\mathbf{X}), \text{LN}(\mathbf{X}))$
$\mathbf{S}' \leftarrow \mathbf{S} + \Delta\mathbf{S}$
$\mathbf{S}' \leftarrow \mathbf{S}' + \text{MLP}_{\text{smb}}(\text{LN}(\mathbf{S}'))$
if SMB-only MHSA is enabled **then**
    $\mathbf{S}' \leftarrow \mathbf{S}' + \text{MHSA}_{\text{smb}}(\text{LN}(\mathbf{S}'))$
end if
for $i = 1, \ldots, N$ **do**
    $m_i^\star \leftarrow \arg\max_m \mathbf{L}_{m,i}$
    $\hat{\mathbf{x}}_i \leftarrow g_\theta([\phi_s(\mathbf{S}'_{m_i^\star}); \phi_p(\mathbf{P}_i)])$
    if $\tau = \texttt{L2}$ **then**
        $\mathbf{r}_i \leftarrow \mathbf{x}_i - \hat{\mathbf{x}}_i$
    else
        $\mathbf{u}_i \leftarrow \hat{\mathbf{x}}_i / \|\hat{\mathbf{x}}_i\|_2$
        $\mathbf{r}_i \leftarrow \mathbf{x}_i - (\mathbf{x}_i^\top \mathbf{u}_i)\mathbf{u}_i$
    end if
    $s_i \leftarrow \|\mathbf{r}_i\|$
end for
$k \leftarrow \max(1, \text{round}(N\rho))$
$\mathcal{I}_{\text{keep}} \leftarrow \text{TopKIndices}(\{s_i\}_{i=1}^N, k)$
$\mathbf{X}_{\text{keep}} \leftarrow \{\mathbf{r}_i : i \in \mathcal{I}_{\text{keep}}\}$
$\Omega \leftarrow \{1, \ldots, N\} \setminus \mathcal{I}_{\text{keep}}$ if dropped-only loss else $\{1, \ldots, N\}$
if $\tau = \texttt{L2}$ **then**
    $\mathcal{L}_{\text{rec}} \leftarrow \frac{1}{|\Omega|} \sum_{i\in\Omega} \frac{\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2}{D}$
else
    sample anchors from Ω
    form positive logits from $(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ and negatives from same-SMB wrong positions and other images
    $\mathcal{L}_{\text{rec}} \leftarrow \text{InfoNCE}(\text{logits})$
end if
return $\mathbf{S}'$, $\mathbf{X}_{\text{keep}}$, $\mathcal{L}_{\text{rec}}$

A Appendix

A.1 Implementation

Algorithm 1 summarizes one compressive layer. The layer first updates a small summary bank by asymmetric cross-attention, then reconstructs each token from its routed summary slot and positional embedding, and finally retains only the tokens with the largest reconstruction residuals.

B Experiments

B.1 Classification Setup

We follow the standard timm ImageNet training recipe, including stochastic depth with rate 0.1, RandAugment (rand-m9-mstd0.5-inc1), bicubic interpolation, color jitter of 0.4, random erasing with probability 0.25 in pixel mode, label smoothing of 0.1, mixup with $\alpha = 0.8$, CutMix with $\alpha = 1.0$, mixup probability 1.0, and mixup/CutMix switching probability 0.5. During evaluation, we use a crop ratio of 0.875 with bicubic interpolation. Optimization is performed with AdamW (learning rate 10^{-3}, weight decay 0.05, $\beta = (0.9, 0.999)$), cosine learning-rate decay to 10^{-5}, gradient clipping at 1.0, automatic mixed precision, and no gradient accumulation. The vanilla models are trained for 300 epochs with 5 warmup epochs, while the compressive models are trained for 500 epochs with

Table 5: ImageNet1k top-1 accuracy and computational cost of the proposed compressive model under various settings. comp-f and comp-r refer to our model using compressive layers that output full vs residual tokens, respectively.

Model	Heads	Dim	Layers	Teacher	SMB	Configuration	Keep ratios	GMACs	Accuracy
Vanilla	6	384	12	–	–	–	–	4.61	80.61
DeiT-S	6	384	12	regnet-y-160	–	–	–	4.61	81.88
Comp-R	6	384	12	Vanilla	16	A1A3A5	0.75,0.25,0.25	2.43	79.61
Comp-F	6	384	12	Vanilla	16	A1A3A5	0.75,0.25,0.25	2.43	79.64
Comp-R	6	384	12	Vanilla	16	A1A3A5	0.75,0.25,0.00	2.33	79.75
Comp-F	6	384	12	Vanilla	32	A1A3A5	0.75,0.25,0.25	2.88	80.44
Comp-R	6	384	12	Vanilla	32	A1A3A5	0.50,0.14,0.00	2.42	79.79
Comp-F	6	384	12	Vanilla	32	A1A3A5	0.50,0.14,0.00	2.42	79.90
Comp-F	6	384	12	regnet-y-160	16	A3	0.00	2.17	79.58
Comp-F	6	384	12	regnet-y-160	8	A4	0.00	2.35	79.26
Comp-F	6	384	12	regnet-y-160	16	A4	0.00	2.55	80.31
Comp-F	6	384	12	regnet-y-160	8	A5	0.00	2.73	79.92
Comp-F	6	384	12	regnet-y-160	16	A5	0.00	2.93	80.79
Comp-F	6	384	12	regnet-y-160	16	A3A5	0.19,0.00	2.36	79.98
Comp-F	6	384	12	regnet-y-160	8	A1A3A5	0.80,0.50,0.50	2.78	80.31
Comp-F	6	384	12	regnet-y-160	8	A1A3A5	0.75,0.25,0.25	2.20	79.26
Comp-F	6	384	12	regnet-y-160	8	A1A3A5	0.75,0.25,0.00	2.11	79.27
Comp-F	6	384	12	regnet-y-160	8	A1A3A5	0.50,0.14,0.00	1.76	78.21
Comp-F	6	384	12	regnet-y-160	16	A1A3A5	0.75,0.25,0.25	2.43	79.99
Comp-F	6	384	12	regnet-y-160	16	A1A3A5	0.75,0.25,0.00	2.33	79.92
Comp-F	6	384	12	regnet-y-160	16	A1A3A5	0.50,0.14,0.00	1.98	78.90
Comp-F	6	384	12	regnet-y-160	16	A1A3A5	0.40,0.10,0.00	1.86	78.73
Comp-F	6	384	12	regnet-y-160	32	A1A3A5	0.75,0.25,0.00	2.78	80.55
Comp-F	6	384	12	regnet-y-160	32	A1A3A5	0.50,0.14,0.00	2.42	80.06
Comp-F	6	384	12	regnet-y-160	32	A1A3A5	0.40,0.10,0.00	2.30	79.69

20 warmup epochs. We also maintain an exponential moving average of the student weights during training (decay 0.99996) and use it for evaluation.

B.1.1 Results

We first train a vanilla DeiT-S-style model without distillation and obtain 80.61% top-1 accuracy on ImageNet. We also train a DeiT-S-style model with RegNet-Y-160 as the teacher, which achieves 81.88% top-1 accuracy. Based on these two backbones, we train two sets of compressive models: one distilled from the vanilla model and one distilled from the stronger RegNet-Y-160 teacher. Full results are given in Table 5. For the placement study, we evaluate a single compressive layer with a keep ratio of 0.0. This setting retains only the SMB tokens and removes all patch tokens after the compressive layer, similar to TokenLearner Ryoo et al. (2021) and STViT Chang et al. (2023).

SMB Partition The color-coded masks exhibit semantic aggregation, suggesting that different SMB tokens tend to absorb distinct types of semantic information. This tendency is more pronounced in the A1A3A5 model than in the A5 model when comparing the tokens entering their first compression layers. This observation suggests that progressive, multi-layer compression may promote more semantically coherent aggregation than single-layer compression.

Ablations We conduct ablation studies to evaluate the contribution of each model component and architectural choice (Table 6). We use the A1A3A5 configuration with keep ratios of 0.75, 0.25, and 0.0 as the default setting for ablations, which achieves 79.92% top-1 accuracy on ImageNet-1K. First, replacing the outputs of the compressive layers with residual tokens instead of original tokens does



Figure 6: Patch-to-SMB assignments in our single-layer A5 classification model with a keep ratio of 0.0 and our three-layer A1A3A5 classification model with keep ratios of 0.75, 0.25, and 0.0. From left to right, the figure shows the patch-to-SMB assignments and corresponding color-coded masks at the single compression layer of the A5 model, followed by the patch-to-SMB assignments for the tokens entering the first, second, and third compression layers of the A1A3A5 model. For example, the visualization at the first layer therefore includes all input patch tokens; this layer subsequently drops 25% of them, and the retained tokens enter the second compression layer. Green numbers identify the SMB tokens assigned to reconstruct the corresponding patches, with patches assigned to the same SMB sharing the same color. Patches without a number have been dropped by preceding compression layers. The examples are randomly selected from the ImageNet validation set.

Table 6: Ablation studies on model components and architectural design choices.

Model	Ablation	Accuracy
A1A3A5-SMB16-0.75, 0.25, 0.0	–	79.92
A1A3A5-SMB16-0.75, 0.25, 0.0	return residual tokens	79.93
A1A3A5-SMB16-0.75, 0.25, 0.0	no decoder	75.95
A1A3A5-SMB16-0.75, 0.25, 0.0	random token dropping	79.21
A1A3A5-SMB16-0.75, 0.25, 0.0	decoder without positional input	79.91
A1A3A5-SMB16-0.75, 0.25, 0.0	late SMB injection	79.70
A5-SMB16-0.0	–	80.79
A5-SMB16-0.0	late SMB injection	79.59

not change accuracy, suggesting that the retained representation is already sufficient for downstream classification.

Next, removing the decoder entirely while keeping the SMB token stream leads to an approximately 4% drop in accuracy, indicating that the reconstruction objective is important for learning useful SMB tokens. When the decoder and reconstruction loss are preserved, replacing residual-norm-based token dropping with random dropping results in a modest 0.7% drop, suggesting that residual norms provide a useful, though not perfect, proxy for token importance. We also examine when the SMB stream is introduced. In our default design, SMB tokens are present from the first layer, which slightly increases compute. As an alternative, we inject learnable SMB tokens only at the first compressive

layer. This causes a 0.2% accuracy drop in the A1A3A5 setting, but a larger 1.2% drop in the A5 setting, where compression begins later in the network.

Finally, we replace the positional embedding input to the decoder with a constant zero vector, effectively learning a per-SMB prototype. This change has little effect on classification accuracy, even though the reconstruction error increases substantially during training. To quantify this effect, we compute the coefficient of determination (R^2) for token predictions in the A1A3A5 model with and without positional embeddings in the decoder. With correct positional embeddings, the R^2 values are 0.4827, 0.4814, and 0.5752 for the three compressive layers, respectively. These values decrease to 0.2741, 0.2669, and 0.3168 when the decoder always receives the zero vector as positional input.

B.2 MLLM

Implementation Details. Following LLaVA-1.5, we use AdamW with a cosine learning-rate schedule. The only modifications are the inserted compressive layers and the additional reconstruction loss, which is applied during both pretraining and fine-tuning with different weights to balance compression and task adaptation. We keep the rest of the training recipe unchanged and train the two modules jointly for 2 epochs. For evaluation, we use greedy decoding with temperature set to 0 to ensure deterministic outputs.

Datasets. We use the 558K image-text alignment dataset and the 665K multimodal instruction dataset released with the backbone Liu et al. (2024a) for pretraining and instruction tuning, respectively. We evaluate our model on seven widely adopted benchmarks, including MMBench Liu et al. (2024c), MME Fu et al. (2023), POPE Li et al. (2023), SQA Lu et al. (2022), VQAText Singh et al. (2019), and VQA Goyal et al. (2017).

Multi-model Understanding We evaluate the proposed compression framework on multimodal reasoning and understanding benchmarks using LLaVA-1.5-13B Liu et al. (2024a) under retained token budgets of 128, 64, and 32. We include ToMe Bolya et al. (2023) as a related baseline, although its performance is relatively limited in the MLLM setting, since it represents an early attempt to bridge ViT compression and multimodal models. For MMBench and MMBench(CN) Liu et al. (2024c), we report dev-set results because test labels are not publicly available and the online evaluation server is no longer accessible.

As shown in Table 7, SMB remains competitive across token budgets while substantially reducing the visual prefix length. It is particularly effective on reasoning-heavy benchmarks such as MMB and MMB-CN, and it continues to preserve most of the baseline capability with 64 or even 32 retained tokens, indicating that the learned compression is robust and scalable. On POPE, the relatively small gap on the adversarial split suggests that token compression has limited additional impact on robustness to semantically plausible absent objects, while the lower recall indicates that the compressed model remains conservative in predicting object presence.

Efficiency. We measure prefill latency on POPE on a single A800 GPU, as reported in Table 8. In the 64-token setting, our method achieves a prefill latency of 37.84 ms per sample, corresponding to a $2.97\times$ speedup over the backbone, while retaining 96% of the backbone performance on POPE. Because compression shortens the visual prefix during both prefill and training, it also yields a comparable reduction in training and fine-tuning time.

B.3 Semantic Partition Map in MLLM

The visualization of SMB partitioning further shows that after two compression layers, the retained visual tokens align well with the keywords in the question prompt, indicating that the kept tokens preserve the fine-grained visual details relevant to downstream reasoning and remain semantically grounded in the task-specific query. Meanwhile, SMB provides complementary global semantic context. Together, keyword-level detail preservation and global semantic coverage help explain the effectiveness of our method under visual self-driven signals, without relying on attention saliency.

Table 7: Comparison of token retention methods on LLaVA-1.5-13B across varying token budgets. Raw scores are reported in the first row and average percentages to the baseline in the final column. Results for SparseVLM and ToMe are sourced from Zhang et al. (2024). Comparisons are best interpreted via relative percentages. The baseline scores 76.4 and 72.4 on the dev splits of MMB and MMB^{CN}, respectively

Method	**VQA**^{v2}	**SQA**^{IMG}	**TextVQA**	**POPE**	**MME**	**MMB**	**MMB**^{CN}	**Avg.**
Vanilla 576 Tokens								
LLaVA-1.5-13B	80.0	71.6	61.3	86.0	1531	67.7	63.5	100%
Retain 128 Tokens								
ToMe(ICLR'23) [1]	-	59.6	49.1	62.8	1343	53.3	-	79.6%
FastV[4](ECCV'24)	75.3	74.2	58.6	75.5	1460	66.1	62.3	96.0%
PDrop[28](CVPR'25)	78.2	73.3	60.2	83.6	1489	67.5	62.8	98.4%
SparseVLM[32](ICML'25)	-	68.6	56.7	85.0	1488	64.5	-	97.7%
PruMerge+[23](ICCV'25)	76.2	73.3	56.1	82.7	1446	66.3	61.2	95.8%
VisionZip[30](CVPR'25)	76.8	73.8	58.9	82.7	1450	67.4	62.5	96.7%
ReConV(Ours)	75.0	70.9	55.3	82.2	1457	74.0	71.5	95.8%
Retain 64 Tokens								
ToMe[1](ICLR'23)	-	50.0	45.3	52.5	1138	43.7	-	67.9%
FastV[4](ECCV'24)	65.3	73.1	53.4	56.9	1246	59.2	55.1	85.8%
PDrop[28](CVPR'25)	70.8	73.1	55.3	66.1	1247	63.1	56.6	88.7%
SparseVLM[32](ICML'25)	-	69.8	53.4	77.5	1373	60.1	-	92.1%
PruMerge+[23](ICCV'25)	72.6	73.5	54.4	75.7	1338	65.0	59.3	92.3%
VisionZip[30](CVPR'25)	73.7	74.2	57.4	75.7	1380	64.9	61.3	93.9%
ReConV(Ours)	72.5	68.5	52.9	82.6	1404	73.1	68.5	92.6%
Retain 32 Tokens								
ReConV(Ours)	70.9	70.6	52.8	82.1	1352	72.6	69.0	92.5%

Table 8: Efficiency analysis on POPE. We report prefill latency per sample, measured on an A800 GPU (40GB). Speedup is computed relative to the Baseline.

Method	Tokens	POPE Accuracy			Prefill Latency	Speedup
		rand	pop	adv	/Sample(ms)↓	(Prefill)↑
Baseline	576	87.1	86.2	84.5	112.25	1.00×
ReConV(Ours)	127	83.3	82.8	80.4	49.63	2.26×
ReConV(Ours)	64	84.5	82.9	80.3	37.84	2.97×
ReConV(Ours)	32	84.8	82.5	78.9	36.55	3.07×

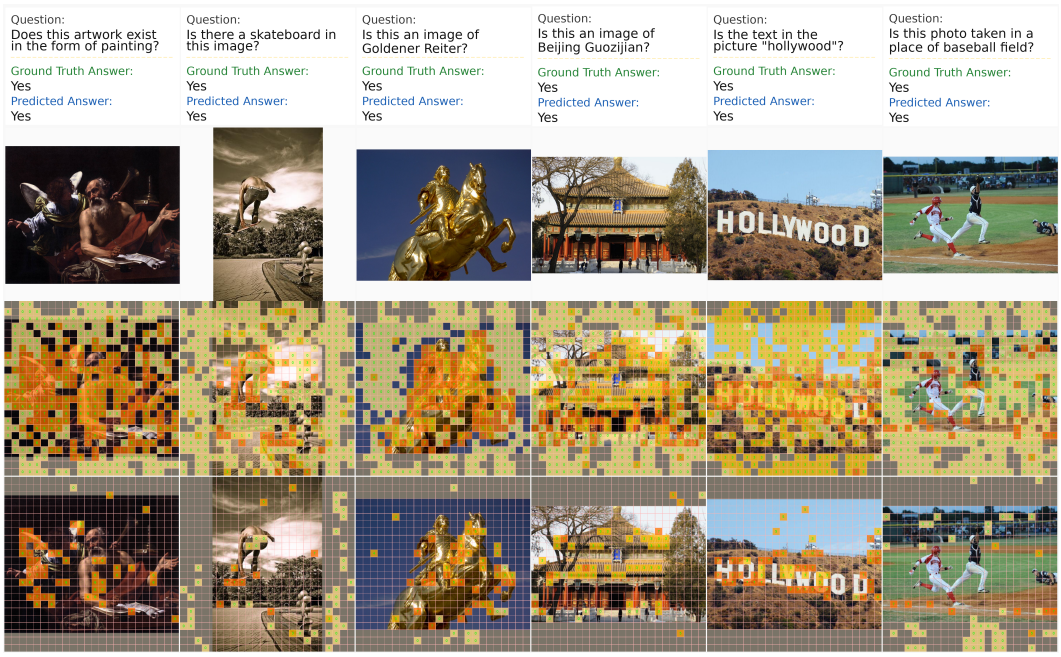

Figure 7: Samples are randomly selected from the MME evaluation set, covering five subtasks from left to right: artwork, existence, landmark, OCR, and scene. The figure visualizes the 64-token setting of our model, which contains 56 kept tokens and 8 SMBs. From top to bottom, the panels show: the question-answer pair, the input image, SMB allocation at the first compression layer, SMB allocation at the second compression layer after the first token-dropping step, and the final SMB allocation after the second token-dropping step. The color from light to dark yellow indicates SMB indices from 0 to 8. The letters on each patch indicate the assigned SMB index, while patches without letters have already been dropped.