

A Technical Appendices and Supplementary Material

A.1 Dataset

We evaluated our method across fifteen benchmarks: twelve for image understanding tasks and three for video understanding tasks, each assessing different aspects of multimodal intelligence.

GQA [20] The GQA benchmark is structured around three parts: scene graphs, questions, and images. It enriches visual content with comprehensive spatial information and object-level attributes. Questions are specifically designed to evaluate models’ ability to comprehend visual scenes and reason about diverse aspects of images.

MMBench [37] The MMBench Benchmark evaluates models through three hierarchical levels of abilities. The first level (L-1) focuses on fundamental perception and reasoning capabilities. The second level (L-2) expands into six distinct sub-abilities, while the third level (L-3) further refines these into 20 specific dimensions. This structure enables a granular and comprehensive assessment of a model’s various capabilities. Additionally, MMB-CN is the Chinese version of the benchmark.

MME [13] MME comprehensively evaluates models’ perceptual and cognitive abilities across 14 subtasks. By employing manually constructed instruction-answer pairs and concise instructions, it effectively minimizes data leakage and ensures fair assessment of model performance.

POPE [30] The POPE benchmark systematically evaluates object hallucination in models through a series of binary questions about object presence in images. Using accuracy, recall, precision, and F1 score as metrics, it precisely measures hallucination levels across different sampling strategies.

ScienceQA [38] ScienceQA encompasses a wide array of domains, including natural, language, and social sciences, with questions hierarchically organized into 26 topics, 127 categories, and 379 skills. Through this comprehensive structure, it offers diverse scientific questions that effectively evaluate multimodal understanding, multi-step reasoning capabilities, and interpretability.

VQA-V2 [15] VQA-V2 evaluates models’ visual perception capabilities through open-ended questions about 265,016 real-world scene images. Each question contains 10 human-annotated ground truth answers, enabling thorough assessment of a model’s ability to interpret and respond to visual queries.

TextVQA [43] TextVQA focuses on the integration of textual information within images. It evaluates a model’s ability to interpret visual elements and embedded text in images through tasks, requiring both visual and textual comprehension to answer questions accurately.

VizWiz [18] The VizWiz benchmark contains over 31,000 visual questions created by blind individuals who photographed scenes with mobile phones while recording spoken questions. Each question includes 10 crowdsourced answers. The dataset presents distinct challenges: lower-quality images from visually impaired photographers, conversational spoken questions, and some questions that remain unanswerable due to content limitations.

SEEDBench [25] SEEDBench contains 19,000 human-annotated multiple-choice questions evaluating models across 12 distinct aspects. It comprehensively assesses capabilities in recognizing spatial and temporal patterns within both images and videos.

MMVet [53] MMVet defines six core abilities (recognition, OCR, knowledge, language generation, spatial awareness, and math) and evaluates models through 16 specific capability combinations. These integrations enable comprehensive assessment of models’ performance across complex scenarios.

LLaVA-Bench [34] LLaVA-Bench categorizes questions into three types: conversational (simple QA), detailed description, and complex reasoning tasks, featuring 24 diverse images with 60 questions spanning indoor and outdoor scenes, memes, paintings, sketches, and more. Each image contains a manually curated detailed description and carefully selected questions, designed to assess model robustness across various prompts.

TGIF-QA [21] TGIF-QA extends the image question answering task to videos with 165,000 question-answer pairs based on GIFs. It introduces four task types: three requiring spatio-temporal reasoning (repetition count, repeating action, state transition) and frame QA answerable from single frames.

MSVD-QA [51] The MSVD-QA benchmark, based on the MSVD dataset, features 1,970 video clips with approximately 50.5K question-answer pairs. It presents open-ended questions across five

categories (what, who, how, when, where) covering diverse video content aspects, serving both video question answering and captioning tasks.

MSRVTT-QA [51] MSRVTT-QA comprises 10K video clips with 243K question-answer pairs, challenging models to integrate visual and temporal elements of video content. Similar to MSVD-QA, it includes five question types, evaluating models’ ability to comprehend complex video content.

A.2 Related Work

Large Vision-Language Models Recently, building on the success of large language models (LLMs) [2, 3, 9, 47], LVLMs [4, 8, 34, 36, 44, 46] have demonstrated impressive performance in multimodal reasoning by bridging visual and linguistic modalities. However, a critical challenge persists in processing images: the quadratic growth of visual tokens as image resolution increases, which leads to prohibitive computational costs during training and inference. High-resolution image models like LLaVA-NeXT [35] and mini-Gemini-HD [28] process thousands of tokens per image, while video models such as VideoLLaVA [31] and VideoPoet [23] demand even more tokens for multiple frames. This highlights the need for more efficient methods to extract information from visual tokens, instead of merely extending their length.

Efficient Large Language Models Efficient inference [11, 10, 24, 33] in LLMs is challenged by their autoregressive nature, where each token prediction depends on the full preceding context. Recently, a variety of token reduction strategies have been proposed to accelerate inference and compress the key-value (KV) cache [19]. For instance, StreamingLLM [49] retains only attention sinks and most recent tokens to shrink the KV cache, while FastGen [14] adaptively manages KV cache based on the behavior of attention head. Heavy-Hitter Oracle (H2O) [56] employs accumulated attention-based scoring to prune key-value pairs during the generation stage. These methods primarily target textual redundancy to improve inference efficiency.

A.3 Evaluation under Additional Settings and Comparison with More Methods

Retain less visual tokens To further validate our method under highly constrained token budgets, we conduct experiments with only 32 visual tokens. As shown in Table 8, our method outperforms existing methods by a notable margin under aggressive token reduction, validating its effectiveness in preserving informative content despite severe compression.

Table 8: Performance of FlowCut on LLaVA-1.5 under 32 visual token setting.

Method	Tokens	GQA	TextVQA	VQA-V2	POPE	SQA	VizWiz	Avg
LLaVA-1.5 7B (upper limit)	576	61.9	58.2	78.5	85.9	69.5	50.0	100%
FastV	32	46.2	51.5	56.0	35.7	68.3	49.1	78.7%
VisionZip	32	51.5	51.7	65.1	68.3	68.2	49.9	88.7%
FlowCut (Ours)	32	52.1	53.0	66.9	69.6	68.7	52.3	90.8%

Compare with more methods We have already compared with several representative methods in the main text. Here, we further include more prior works for comparison: PruMerge [41] clusters the visual tokens via k-nearest merging algorithm and merges them. ST³ [57] prunes low-importance tokens progressively across layers using token attention maps. VTW [32] removes unimportant visual tokens based on attention score at specific layers of the LLM. RCom [7] merges unimportant patch tokens based on their similarity to the CLS token and the text tokens. FasterVLM [54] prunes vision tokens based on CLS attention scores at vision encoder’s output layer.

These methods all rely on single-layer, single-metric scores (e.g., attention or similarity) to assess token importance. In contrast, our method employs multiple metrics and aggregates scores across layers using an EMD-based strategy, which helps mitigate single-metric bias and layer bias. We also discuss our differences with CrossGET [42] here. CrossGET injects learnable cross tokens to capture cross-modal information and merges less important tokens based on cosine similarity. In contrast, our method leverages CLS attention, performs multi-layer and multi-metric evaluation, and applies dynamic layer-wise pruning. Moreover, CrossGET requires re-training due to its learnable cross tokens while our approach is entirely training-free.

As shown in Table 9, our method better preserves performance under similar FLOPs, achieving consistently stronger results across a wide range of benchmarks. Note that for ST³, as the code is not publicly available, we follow their benchmark protocol for fair comparison.

Table 9: Comparison of FlowCut on LLaVA-1.5-7B with more methods.

Method	TFLOPs↓	TextVQA	VQA-V2	POPE	MMB	SQA	Metric	ST ³ [57]	FlowCut (Ours)
LLaVA-1.5 7B	8.82	58.2	78.5	85.9	64.7	69.5	AI2D	55.4	55.8
PruMerge [41]	1.95	53.5	65.9	70.7	56.8	68.5	SQA	68.9	69.2
VTW [32] (K=5)	2.01	8.1	42.7	46.0	21.3	65.3	MMMU	35.3	36.2
RCom [7]	1.96	55.5	70.4	72.0	57.9	69.0	MMB	63.8	64.3
FasterVLM [54]	1.97	55.2	71.9	76.1	60.4	68.9	POPE	85.2	86.1
FlowCut (Ours)	1.95	55.6	72.8	80.2	60.8	69.1	TFLOPs	4.27	4.25

A.4 More Sensitivity Analyses of Hyperparameters

Cumulative importance mechanism’s weighting coefficients We default to a setting of 0.5 for historical scores and 0.5 for current scores. As shown in Table 10, we further conduct a sensitivity analysis. When only the current score is used (i.e., without the cumulative strategy), the POPE performance is 83.8. The best performance is achieved at 0.5:0.5 and 0.6:0.4, while all other settings still outperform the non-cumulative baseline. This demonstrates that our method is robust to changes in the weighting coefficients and validates the effectiveness of the cumulative strategy.

Table 10: Sensitivity analysis of cumulative importance mechanism’s weighting coefficients.

history:current	0:10 (w/o cumulative strategy)	1:9	2:8	3:7	4:6	5:5	6:4	7:3	8:2	9:1
POPE benchmark	83.8 (baseline)	84.6	84.8	84.6	85.0	85.2	85.2	85.1	85.0	84.7

Pruning frequency For pruning frequency, we default to pruning every two layers with dynamic prune ratios. As shown in Table 11: 1) Pruning every two layers yields the best performance. Performance decreases as the interval increases, and pruning every layer (n=1) performs slightly worse than n=2, which we attribute to the fact that the first pruning step under n=1 cannot leverage historical scores and must rely solely on the current layer. 2) Multi-layer pruning outperforms single-layer pruning (i.e., only pruning at the last layer), further demonstrating both robustness to this hyperparameter and the benefit of pruning redundancies as they emerge. 3) These results confirm that our method is not sensitive to the prune frequency, highlighting its robustness.

Table 11: Sensitivity analysis of pruning frequency.

Pruning each n layer	n=1	n=2	n=3	n=4	n=6	n=8	n=12	only prune at last layer
POPE benchmark	84.9	85.2	84.6	84.6	84.3	84.3	84.0	83.9 (baseline)

A.5 Pseudocode

Here, we provide the pseudocode of our method. And note that method is indeed compatible with FlashAttention because it does not require the full $N \times N$ attention map. Instead, it only needs the $1 \times N$ attention value of a single CLS token or global token. When using FlashAttention, we can obtain the necessary attention values with a small, separate computation: 1) For models with a CLS token, we separately compute its $1 \times N$ attention map with the patch tokens; 2) For models without a CLS token, we derive a global token (by averaging patch tokens) and then compute its $1 \times N$ attention map. This targeted computation is highly efficient, with a complexity of just $O(n)$, adding virtually no overhead. This allows our method and FlashAttention to be used together effectively.

Algorithm 1 Pseudocode for FlowCut

```
# Input: current layer's hidden_states and QKV
# Output: selected tokens for current layer
# QKV shape: [B, N, D] (batch size, sequence length, embedding dim)
# If the architecture without CLS token, use a globally pooled token as a substitute

if architecture_without_CLS: cls_token = output.hidden_states.mean(dim=1) # [B,D]
else: cls_token = output.hidden_states[:,0] # [B,D]

patch_token = output.hidden_states[:,1:] # [B,N-1,D]

Q_cls = Q[:, 0] # [B, D], first token is CLS token
K_patch = K[:, 1:] # [B, N-1, D], exclude CLS token

# Compute CLS-to-token attention (O(N) complexity, compatible with FlashAttention)
cls_attn = softmax(dot(Q_cls, K_patch.transpose(-1, -2)) / sqrt(D)) # [B, N-1]

# Compute semantic similarity to evaluate token-level informativeness
V_cls = V[:, 0] # [B, D]
V_patch = V[:, 1:] # [B, N-1, D]
semantic_attn = softmax(dot(V_cls, V_patch.transpose(-1, -2)) / sqrt(D)) # [B, N-1]

# Compute value norm as a proxy for token information capacity
v_score = L1_norm(V_patch) # [B, N-1]

# Fuse multiple importance criteria
importance = (normalize(cls_attn) + normalize(semantic_attn)) * v_score # [B, N-1]

# Accumulate importance across layers (EMA-like smoothing)
cumulative_score = 0.5 * cumulative_score + 0.5 * importance

# Estimate pruning ratio based on CLS attention entropy
entropy = compute_entropy(cls_attn) # [B]
entropy_ratio = entropy / log(N)
prune_ratio = (N - target_num) / sqrt(remain_layer) * (1 - entropy_ratio ** 2)

# Select top-K tokens based on cumulative importance scores
K_keep = round((1 - prune_ratio) * N)
keep_idx = topk(cumulative_score, K=K_keep) # [B, K_keep]

# Retain tokens
if not final_step: tokens = concat(cls_token, filter(patch_tokens, keep_idx))
elif final_step: tokens = filter(patch_tokens, keep_idx)
```

A.6 Broader Impact

Our research rethink the essence of visual token reduction from a fundamental information flow perspective. Through systemical analysis of information flow patterns, we reveal that redundancy as a natural consequence of asymmetric attention. Through comprehensive analysis, we demonstrate the limitations of single-layer, single-criterion metrics in identifying redundancy, and propose FlowCut, an information-flow-aware pruning framework that aligns pruning decisions with the inherent dynamic information flow of tokens. In some cases, we observe that token pruning, by removing redundant and distracting tokens, produces more accurate answers than the non-pruned baseline. This suggests that token pruning may hold promise for mitigating hallucinations in multimodal models. Therefore, it is worth exploring in future work why token pruning helps reduce hallucinations, and how we can better leverage efficient techniques—such as token pruning and token merging—not only to accelerate inference but also to suppress hallucinations.

A.7 More Visualization Results

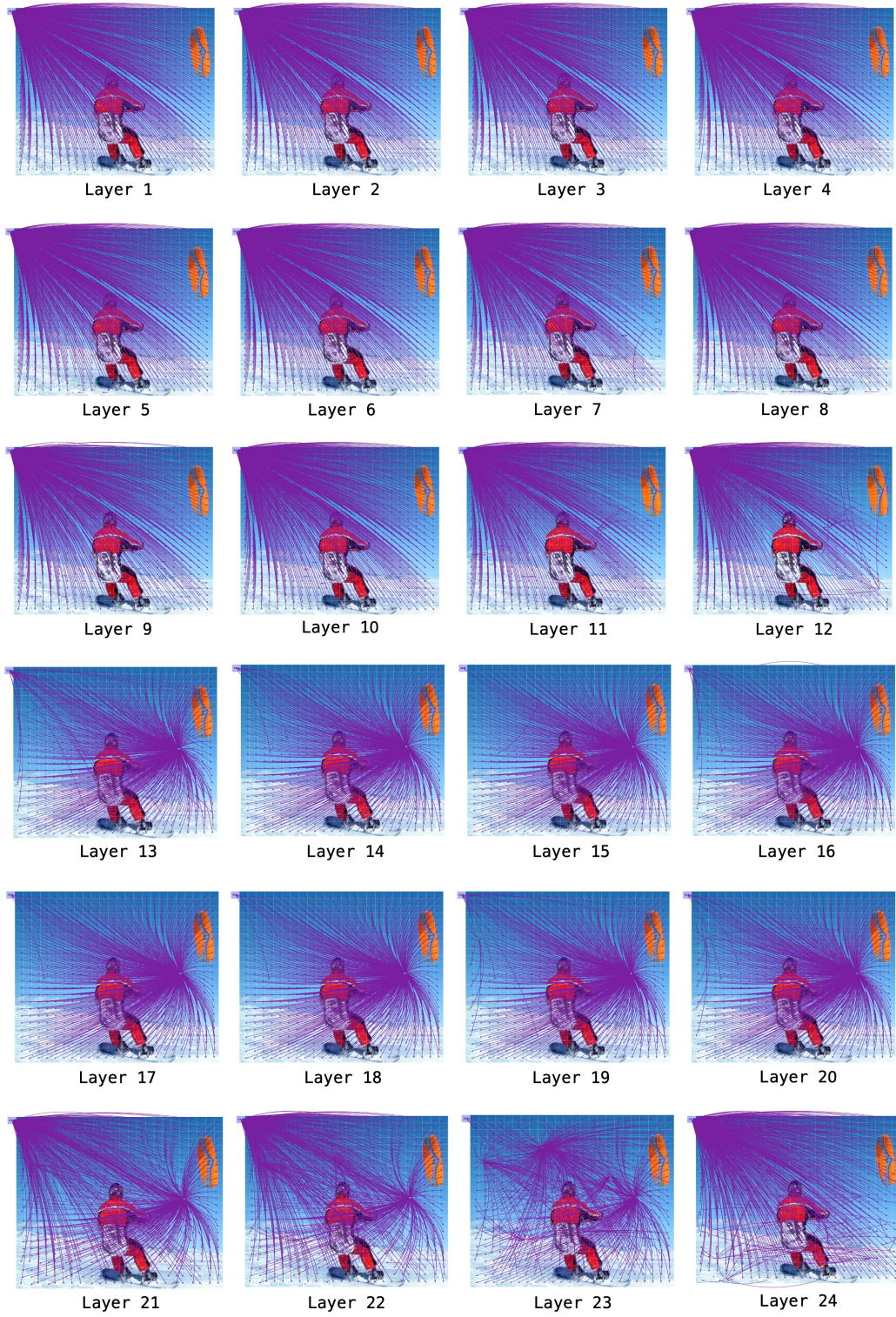


Figure 9: Visualization of information inflow main streamline.



Figure 10: Visualization of information outflow main streamline.

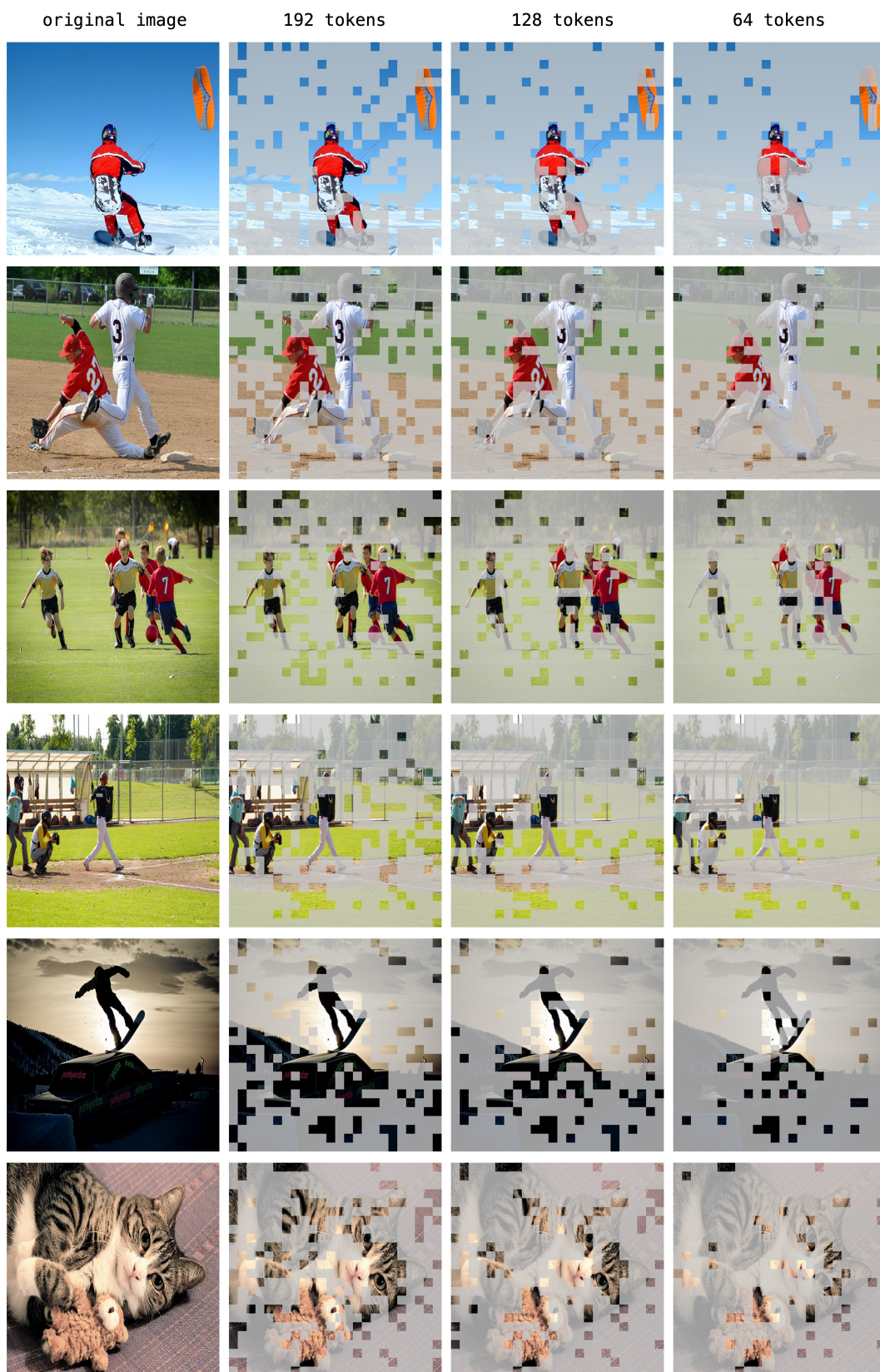


Figure 11: Sparsification Visualization examples of FlowCut on LLaVA-1.5-7B.