Vision-Language Model Token Pruning

2026. 02. 13 || 2026년 동계 세미나

Presented By
문승훈

1) FlashVID: Efficient Video Large Language Models via Training-free Tree-based Spatiotemporal Token Merging | OpenReview
2) HUANG, Yihong, et al. N\" uwa: Mending the Spatial Integrity Torn by VLM Token Pruning. *arXiv preprint arXiv:2602.02951*, 2026.

Outline

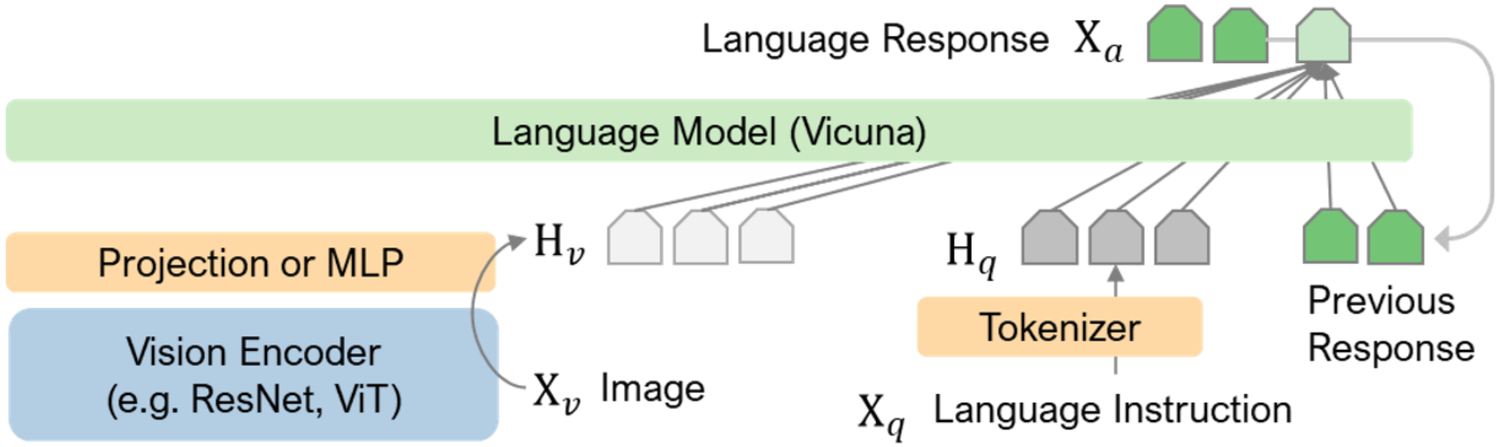
- Background
  - VLM inference bottleneck and visual token redundancy
  - Token compression methods

- FlashVID: Efficient Video Large Language Models via Training-free Tree-based Spatiotemporal Token Merging[1)]
  - ICLR 2026 Oral

- Nüwa: Mending the Spatial Integrity Torn by VLM Token Pruning[2)]
  - ICLR 2026

VLM Inference Bottleneck and Visual Token Redundancy

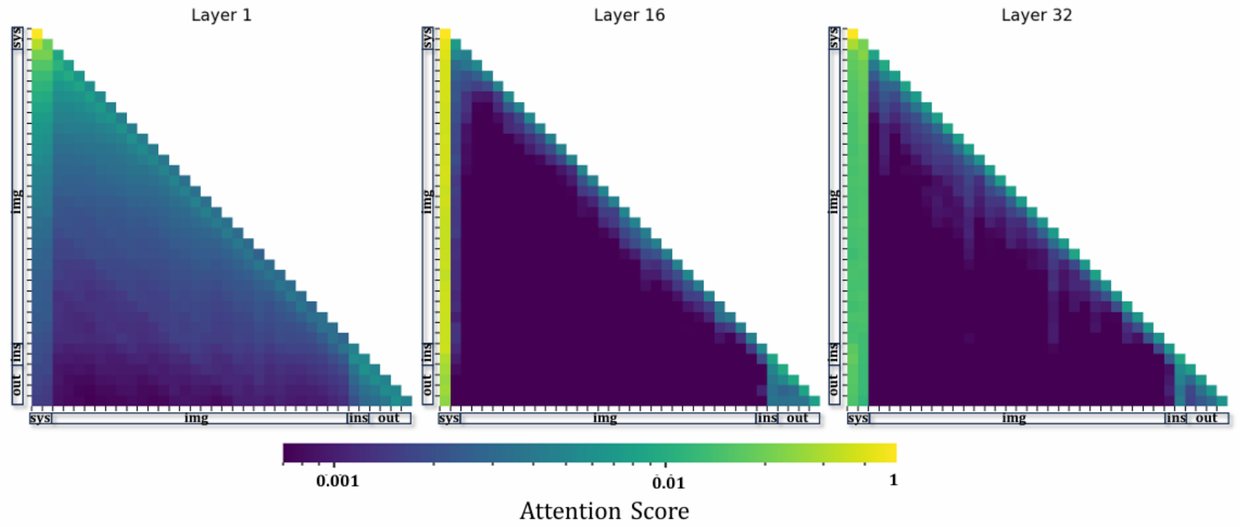
- VLM Inference Bottleneck
  - Transformer의 quadratic complexity
    - Self-attention 연산량은 입력 토큰 수의 제곱에 비례해 고해상도 이미지나 비디오를 입력으로 받을 때 bottleneck 심화
  - KV cache overhead
    - 시각 토큰이 많을수록 KV cache가 차지하는 메모리 점유율이 높아져 throughput이 저하됨
  - Prompt length
    - LLM이 수용 가능한 context window 내에서 시각 토큰이 차지하는 비중이 너무 커지면 복잡한 prompt를 처리하기 어려움



VLM Framework (LLaVA)

VLM Inference Bottleneck and Visual Token Redundancy

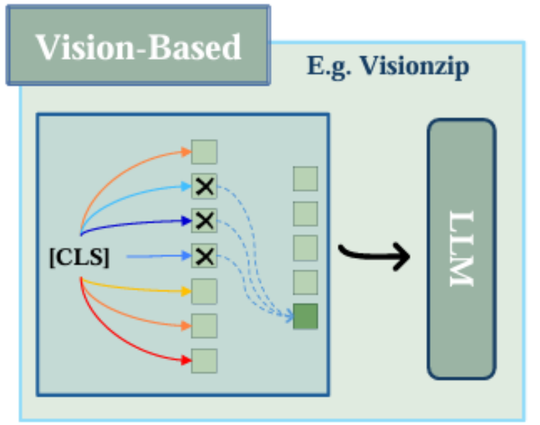
- Visual Token Redundancy
  - Spatial redundancy
    - 이미지의 인접한 patch들은 유사한 feature vector를 가져 모든 patch가 고유한 정보를 가지고 있지 않음
  - Temporal redundancy
    - 비디오의 경우 인접한 프레임 간의 변화가 적어 특정 영역이나 객체가 여러 프레임에 걸쳐 중복된 정보 제공
  - Semantic density
    - 텍스트 토큰에 비해 시각 토큰은 정보 밀도가 낮아 상당수의 토큰을 제거해도 전체적인 context 유지 가능



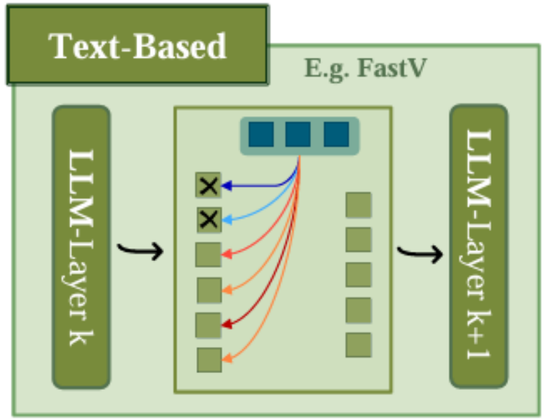
Attention Maps During the Decoding Process for LLaVA1.5-7B

Token Compression Methods

- Before-LLM vs. Inner-LLM
  - Before-LLM (Vision-based)
    - Vision encoder 출력 직후 또는 LLM으로 전달해주는 projector 단계에서 동작
    - 입력 데이터 전체의 visual feature만을 보고 redundancy 제거
    - LLM의 첫 번째 layer부터 token 수가 줄어든 상태이기 때문에 경량화 효과가 제일 큼
    - Text query를 보기 전에 수행되기 때문에 query-agnostic이라고도 함
  - Inner-LLM (Text-based)
    - LLM의 내부 intermediate layer에서 동작
    - Attention 등 LLM 내부에서 텍스트 토큰과 시각 토큰이 상호작용한 결과를 사용해 보존할 토큰을 동적으로 선택
    - 답변에 직접적으로 필요한 정보를 정확하게 선별 가능
    - Text query를 참고하여 수행되기 때문에 query-aware라고도 함



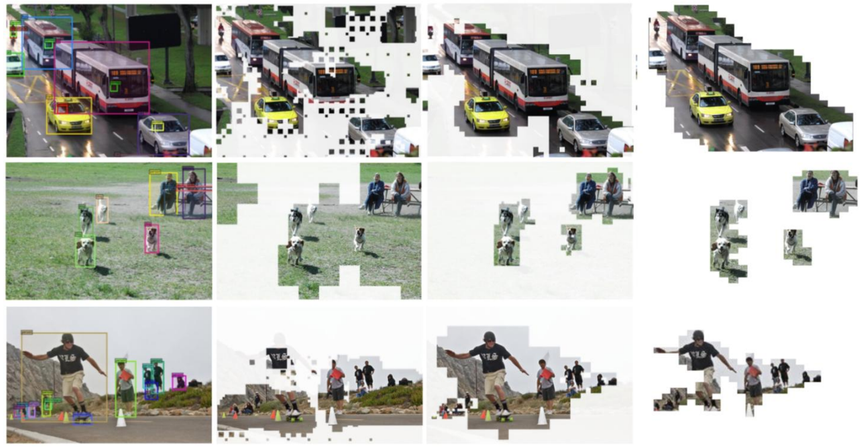
Before-LLM



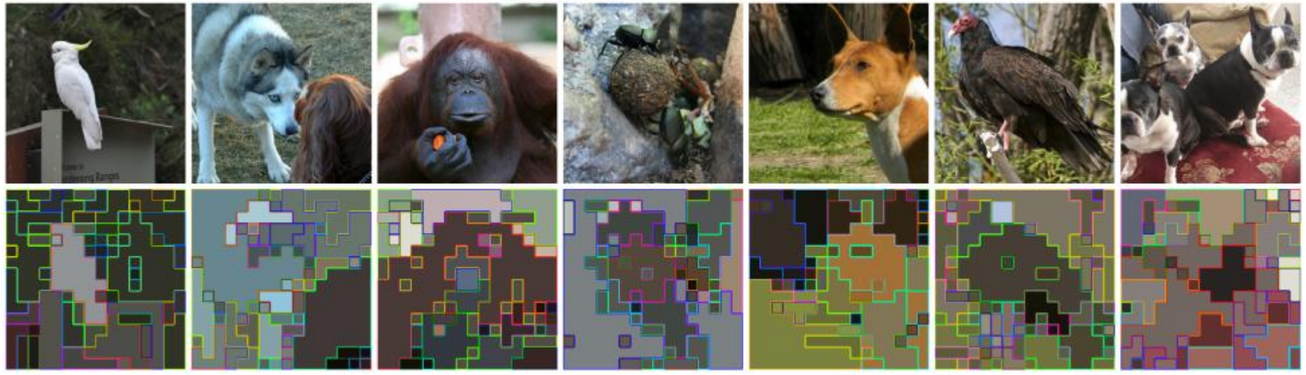
After-LLM

Token Compression Methods

- Pruning vs. Merging
  - Token pruning
    - 덜 중요한 토큰을 식별해 완전히 삭제해 즉각적으로 복잡도를 최적화하는 방식
    - 계산 오버헤드가 적지만 한 번 버려진 토큰의 정보는 복구할 수 없음
  - Token merging
    - 유사한 특징을 갖는 토큰들을 하나로 합쳐 토큰의 총 개수를 줄이는 방식
    - 데이터를 압축하기 때문에 정보 손실을 최소화해 전체적인 맥락을 보존할 수 있음



Token Pruning

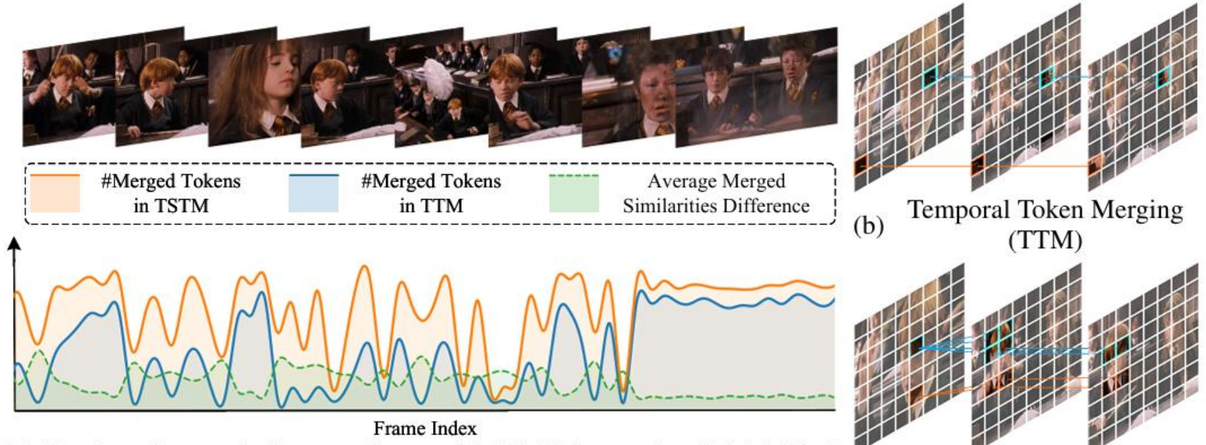


Token Pruning

FlashVID: Efficient Video Large Language Models via Training-free Tree-based Spatiotemporal Token Merging[1)]

Introduction

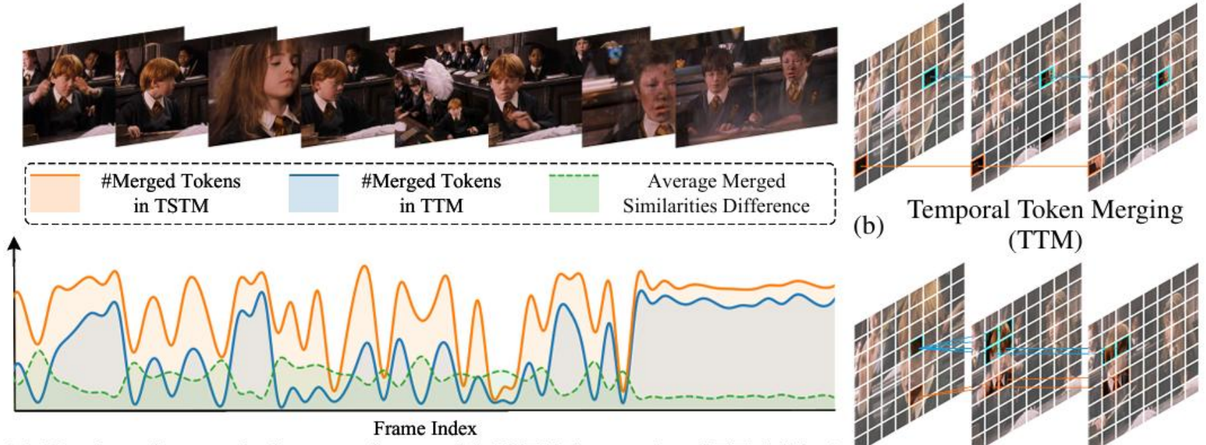
- Motivation
  - 기존 VLLM acceleration 방식은 비디오의 spatiotemporal correlation을 충분히 반영하지 못함
    - VLLM은 수백 개의 프레임에서 발생하는 막대한 시각 토큰을 처리해야 함
      - LLM 내부의 self-attention과 KV cache 사용량을 폭증시키는 요인이 됨
    - 기존 방식은 spatial 및 temporal 차원을 독립 처리하기 때문에 비디오 데이터가 가진 spatiotemporal correlation을 놓침
      - Spatial compression: 프레임 내부의 redundancy만 제거
      - Temporal compression: 특정 프레임을 sampling해 제거



Comparison of Spatiotemporal Redundancy Compression

Introduction

- Motivation
  - 프레임 간 이동하는 객체를 고려한 spatiotemporal 통합 compression과 핵심 정보 선별이 필수
    - 비디오의 dynamic 특성
      - 객체는 시간에 따라 position, scale, orientation 등이 계속 변화함
      - 따라서 동일한 위치에 있는 토큰들은 유사한 정보를 담고 있다고 가정해 여러 프레임의 fixed location의 토큰을 합치는 기존 방식 (TTM)은 실제로는 관련 없는 정보를 섞게 되어 노이즈를 유발함
    - Spatiotemporal 차원을 모두 고려해 객체의 움직임을 따라 adaptive하게 토큰을 compress하는 방법 필요

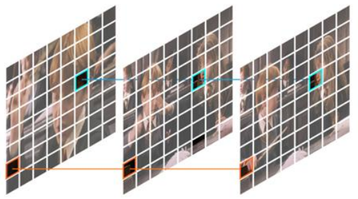


(a) Number of merged tokens per frame with TSTM (orange) and TTM (blue) under the same threshold, with average merging similarity differences between TSTM and TTM shown in green.

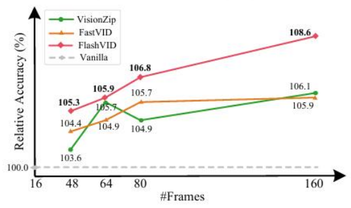
Comparison of Spatiotemporal Redundancy Compression

Introduction

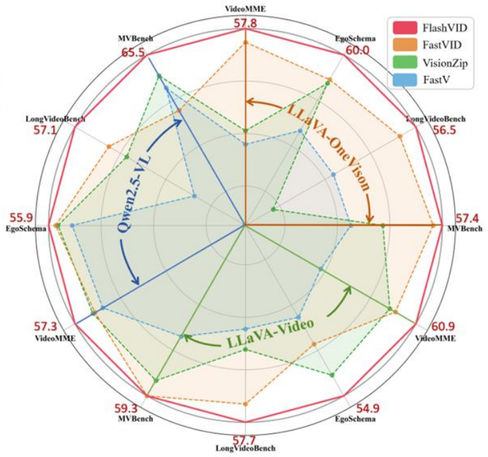
- FlashVID
  - 먼저 중요한 것을 select하고 그 다음 유사한 것들을 계층적으로 merging하는 2-stage 전략
    - ADTS (Attention and Diversity-based Token Selection)
      - 비디오의 raw features는 noisy하고 배경 등 중요하지 않은 정보가 큰 비중을 차지해 바로 압축할 경우 성능 저하 발생
      - 따라서 각 프레임에서 가장 대표성 있는 토큰을 우선적으로 선별
      - TSTM 단계로 넘어가기 전 정보 밀도가 높은 토큰들만 남겨서 token merging의 효율성과 정확도를 높임
    - TSTM (Tree-based Spatiotemporal Token Merging)
      - ADTS로 걸러진 토큰들 사이의 spatiotemporal redundancy를 계층적으로 모델링해 합침
      - 비디오 데이터를 tree 구조로 형상화해 프레임 내부 (spatial)와 사이 (temporal)의 관계를 동시에 고려
      - 비디오의 동적인 특성에 맞춰 adaptive하게 token merging 수행



(a) Temporal Token Merging (TTM)



(b) Process more video frames

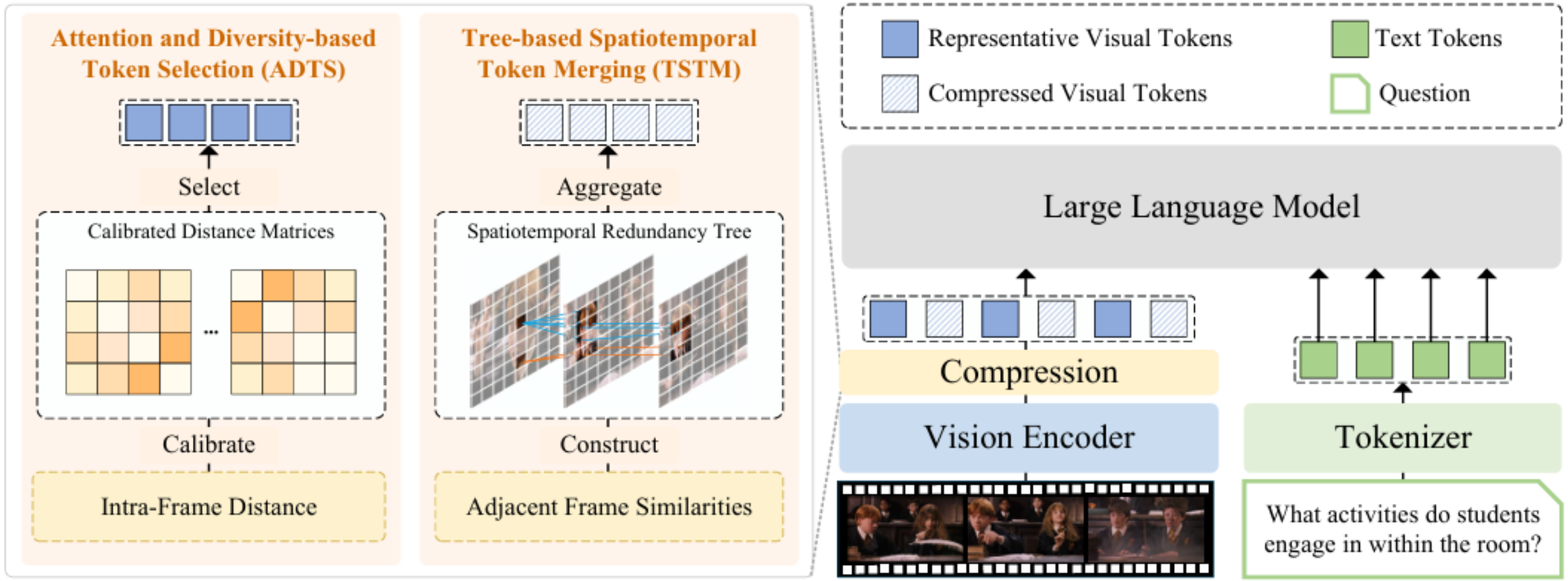


(c) FlashVID achieves SOTA performance

Performance of FlashVID

Methodology

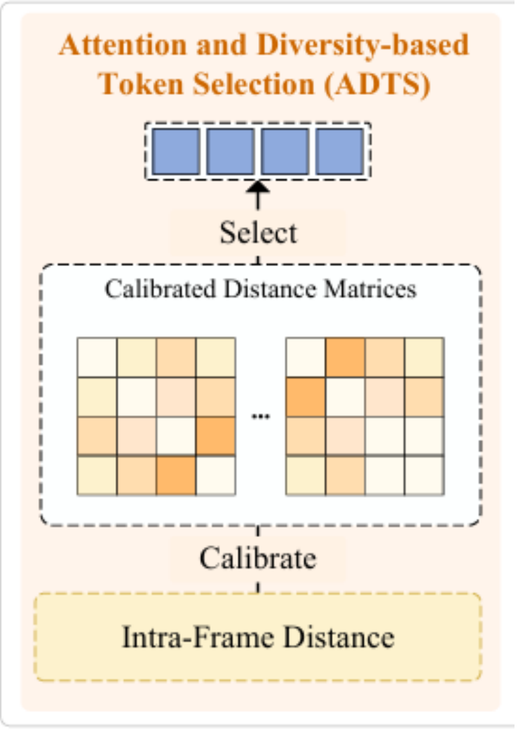
- FlashVID Overview
  - ADTS를 통해 informative and diverse한 토큰을 선택해 robust video representation 구현
  - TSTM을 통해 중요한 시각 정보를 보존하면서 spatiotemporal redundancy 최소화



FlashVID

Methodology

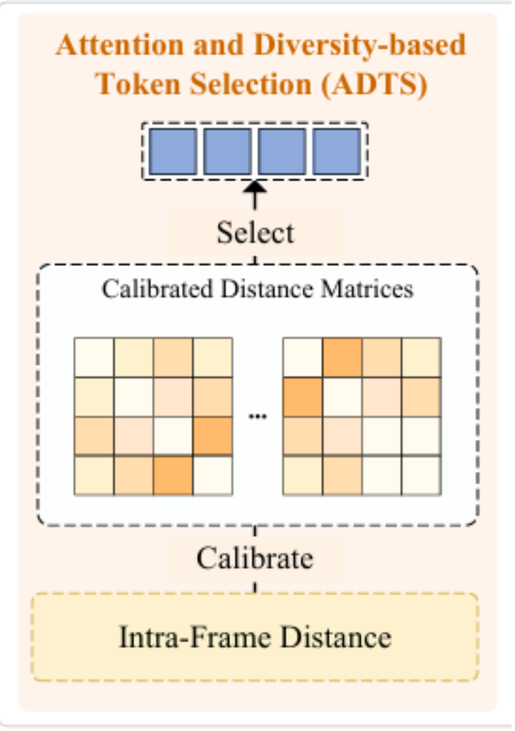
- ADTS
  - TSTM이 수행되기 전 각 프레임 내에서 가장 informative한 토큰을 선별하는 전처리 단계
    - 단순히 attention score가 높은 토큰만 선택하면 redundancy가 완전히 해소되지 않음
    - 또한 diversity만 추구하면 outlier와 같은 노이즈가 선택되는 한계점 존재
    - 따라서 [CLS] attention과 event relevance로 보정된 Max-Min Diversity Problem (MMDP) 제안
  - Step 1: Frame-wise feature dissimilarity $D^{(f)}$ at frame f
    - 프레임 내에서 토큰들이 서로 얼마나 다른지 측정해 중복되지 않은 다양한 시각 정보 포착
      - $D^{(f)} = 1 - \cos\left(E_v^{(f)}, E_v^{(f)}\right)$
      - 값이 클수록 두 토큰 간 정보가 상이함을 의미
  - Step 2: Importance calibration (1/2)
    - [CLS] attention calibration $A_{[CLS]} \in R^{F \times N_v}$
      - Vision encoder가 어디를 주목하는지 반영, $A = \text{Softmax}\left(QK^T/\sqrt{d}\right) \in R^{F \times N_v \times N_v}$
      - 각 토큰이 프레임 내에서 받는 A를 평균내어 계산해 $A_{[CLS]}$ 유도



ADTS

Methodology

- ADTS
  - Step 2: Importance calibration (2/2)
    - Event relevance calibration $\overline{S}_e$
      - 해당 토큰이 비디오의 전체적인 context와 얼마나 관련이 있는지 수치화
      - $f_v = \mathrm{GAP}(E_v) \in R^{F \times d}$ (GAP: Global Average Pooling)
      - 각 토큰과 프레임 임베딩 간의 유사도를 계산해 $\overline{S}_e$ 산출
      - $\overline{S}_e = \frac{1}{F}\sum_{i=1}^{F}(E_v \cdot f_v^T)[:, i] \in R^{F \times N_v}$
  - Step 3: Calibrated MMDP optimization
    - Diversity ($D^{(f)}$)와 importance ($A_{[CLS]}$, $\overline{S}_e$)를 결합해 MMDP의 solution 계산
      - NP-hard 문제인 MMDP를 Greedy algorithm을 사용해 approximate solution 도출
    - 이를 통해 프레임 내에서 서로 diverse하면서도 의미적으로 salient한 토큰만을 선별
    - 선별한 토큰은 다음 단계인 TSTM으로 전달



ADTS

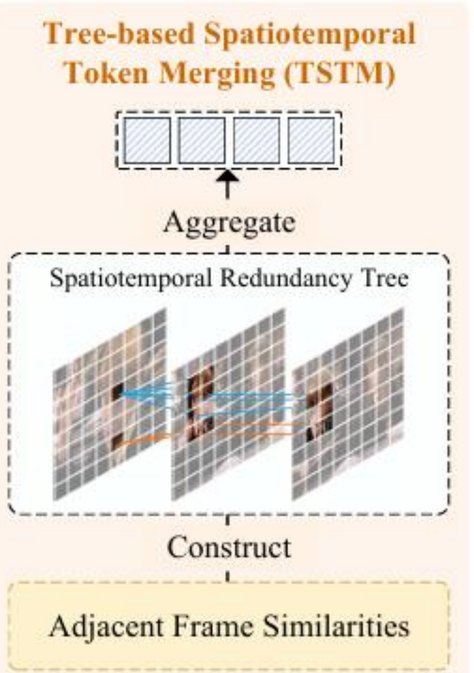
Methodology

- TSTM
  - 비디오 내에서 시간의 흐름에 따라 발생하는 redundancy를 압축하는 알고리즘
    - 기존 방법들은 단순히 같은 좌표에 있는 토큰들을 temporal 축으로 병합하는 구조
    - 이로 인해 물체가 움직일 때 정보를 잃어버리는 문제 발생
    - 따라서 내용이 유사한 토큰들을 찾아 연결하는 tree 구조인 TSTM 제안
  - Step 1: Spatiotemporal redundancy tree 구축 (1/2)
    - Temporal 축을 따라 이동하거나 변형되는 객체의 토큰들을 하나의 branch로 연결
    - Video features의 인접한 두 프레임 f와 $f+1$ 사이의 cosine similarity 계산

$$S^{(f)} = \cos\left(E_v^{(f)}, E_v^{(f+1)}\right) \in R^{N_v \times N_v}$$

$S^{(f)}(j,k)$로 f 프레임의 j 토큰과 $f+1$ 프레임의 k 토큰이 얼마나 유사한지 나타냄

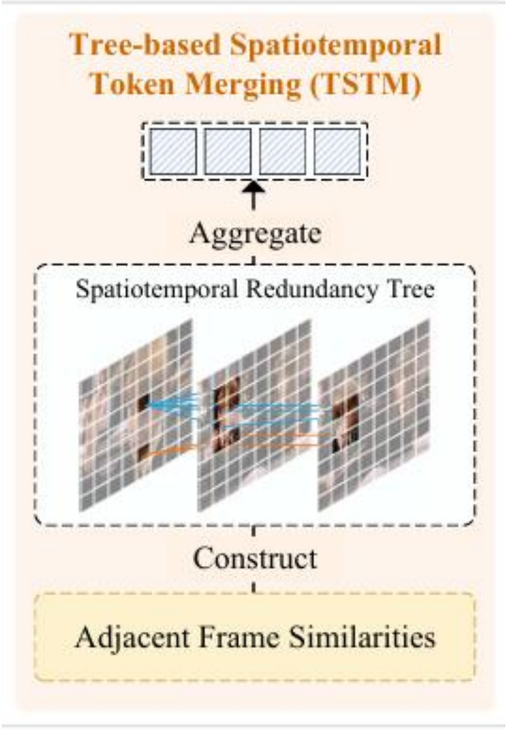
현재 프레임의 특정 토큰이 다음 프레임의 어떤 위치로 이동했는지 확인



TSTM

Methodology

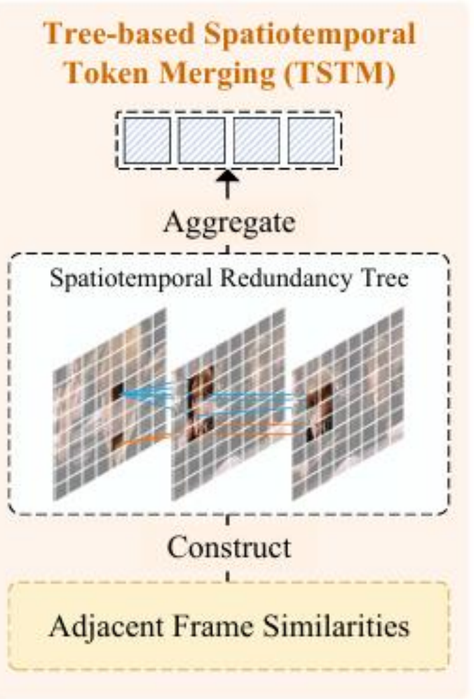
- TSTM
  - Step 1: Spatiotemporal redundancy tree 구축 (2/2)
    - 계산된 similarity matrix를 바탕으로 tree 형성
      - 프레임 f의 각 토큰에 대해 이전 프레임에서 가장 유사도가 높은 토큰 p^* 탐색
      - $p^* \leftarrow \underset{p \in \mathcal{R}^{(f-1)}}{\operatorname{argmax}} \operatorname{sim}(r_i^f, p)$
    - 이때 단순히 가장 비슷한 것을 연결하는 것이 아니라 유사도 값이 사전 정의된 merging threshold T_τ 이상일 때만 연결
      - 이 과정을 전체 프레임에 대해 수행
    - 결과적으로 temporal 축으로 유사한 토큰들이 연결된 tree 구조가 구축됨
    - Tree의 각 node는 서로 다른 프레임에 존재하지만 의미적으로 유사한 객체를 나타냄



TSTM

Methodology

- TSTM
  - Step 2: Spatiotemporal redundancy compression
    - Tree가 구축된 후 하나의 tree에 묶인 토큰들은 같은 정보를 가진 것으로 간주해 merging 수행
      - 각 tree에 포함된 모든 토큰들을 mean pooling으로 aggregate해 하나의 대표 토큰으로 변환
      - 즉 tree에 속한 모든 토큰 벡터들의 평균값으로 token merging 수행
      - 이를 통해 좌표가 아닌 feature similarity를 기준으로 정보 손실을 최소화하면서 압축
    - 결과적으로 여러 프레임에 걸쳐 움직이는 동일 객체가 여러 토큰으로 흩어져 있던 것을 TSTM을 통해 단 하나의 대표 토큰으로 압축
- Tree 구조의 실제 구현
  - Cosine similarity 기준 가장 비슷한 이전 프레임 토큰의 인덱스를 부모 node로 탐색
    - 해당 방식으로 프레임끼리 유사한 토큰의 인덱스를 묶는 구조
    - 마지막 프레임에서 첫 프레임 방향으로 backward aggregation 수행
      - 유사도가 사전 정의된 temporal threshold보다 높을 때만 merge
      - Backward 방향으로 토큰을 더하기만 하다가 더 이상 이전 프레임으로 병합되지 않을 때 나눔



TSTM

Experiments

Method	Retention Ratio R	VideoMME				EgoSchema		LongVideo Bench	MVBench	Avg.	
		Short	Medium	Long	Overall	Subset	Total			Score	Rel. Acc (%)
LLaVA-OneVision											
Vanilla	100%	69.9	56.7	48.9	58.5	62.2	60.3	56.6	58.3	58.4	100.0
FastV	25%	68.1	54.7	46.8	56.5	60.4	57.8	55.4	56.4	56.5	96.7
VisionZip	25%	68.8	57.3	48.2	58.1	63.0	60.4	56.4	57.8	58.2	99.7
PruneVID	25%	67.3	54.8	47.2	56.4	61.0	58.1	55.4	56.8	56.7	97.1
FastVID	25%	69.9	56.3	47.4	57.9	61.2	59.5	55.9	58.1	57.8	99.0
FlashVID	25%	**71.2**	**57.0**	**49.3**	**59.2**	**63.4**	**60.4**	**56.8**	**58.0**	**58.6**	**100.3**
FastV	20%	66.3	53.9	46.9	55.7	60.6	57.6	56.0	56.0	56.3	96.4
VisionZip	20%	68.6	**57.0**	48.3	58.0	62.0	60.0	55.4	57.6	57.7	98.8
PruneVID	20%	67.2	53.9	48.2	56.4	**63.2**	**60.2**	55.2	56.2	57.0	97.6
FastVID	20%	69.9	56.3	47.4	57.9	61.2	59.5	55.9	58.1	57.9	99.1
FlashVID	20%	**70.1**	55.4	**48.9**	**58.2**	63.0	60.1	**58.5**	**58.2**	**58.7**	**100.5**
FastV	15%	64.6	54.0	45.3	54.6	59.8	56.6	54.8	55.0	55.2	94.5
VisionZip	15%	63.8	54.6	48.3	55.6	62.8	60.0	54.1	53.5	55.8	95.5
FastVID	15%	**69.7**	55.8	47.7	57.7	58.8	58.9	56.7	**58.2**	57.9	99.1
PruneVID	15%	67.2	52.8	46.7	56.1	61.6	57.7	54.5	55.1	55.7	95.4
FlashVID	15%	69.6	**56.0**	**48.9**	**58.2**	**62.8**	**60.4**	**57.5**	57.9	**58.5**	**100.2**
FastV	10%	60.9	52.2	44.9	52.7	59.0	56.0	52.4	53.4	53.6	91.8
VisionZip	10%	60.3	52.9	46.7	53.3	61.6	58.5	49.4	54.8	54.0	92.5
PruneVID	10%	65.9	52.8	45.6	54.7	60.0	57.2	54.0	53.7	54.9	94.0
FastVID	10%	**68.1**	55.7	47.8	57.2	58.8	58.7	55.7	57.0	57.1	97.8
FlashVID	10%	67.3	**57.1**	**49.0**	**57.8**	**62.4**	**60.0**	**56.5**	**57.4**	**57.9**	**99.1**
LLaVA-Video											
Vanilla	100%	77.0	62.1	53.3	64.2	59.4	57.3	59.5	61.9	60.7	100.0
FastV	20%	69.3	58.3	49.9	59.2	54.8	54.1	56.0	58.4	56.9	93.7
VisionZip	20%	72.3	59.6	53.3	61.7	**59.0**	56.4	58.0	59.8	59.0	97.2
FastVID	20%	**74.6**	**60.8**	52.3	**62.6**	57.0	55.0	57.1	**60.2**	58.7	96.7
FlashVID	20%	74.1	60.0	**52.3**	62.2	58.4	**56.4**	**58.7**	59.8	**59.3**	**97.7**
FastV	10%	64.3	53.8	49.2	55.8	50.6	51.1	53.6	56.2	54.2	89.3
VisionZip	10%	69.4	57.9	51.2	59.5	54.4	53.9	54.5	58.5	56.6	93.2
FastVID	10%	71.8	57.3	50.2	59.8	54.8	52.4	56.9	59.3	57.1	94.1
FlashVID	10%	**72.2**	**59.1**	**51.2**	**60.9**	**57.2**	**54.9**	**57.7**	**59.3**	**58.2**	**95.9**

Comparison on LLaVA-OneVision and LLaVA-Video

Method	Retention Ratio R	VideoMME				EgoSchema		LongVideo Bench	MVBench	Avg.	
		Short	Medium	Long	Overall	Subset	Total			Score	Rel. Acc (%)
Vanilla	100%	72.6	61.4	49.9	61.3	60.2	58.3	58.9	68.0	61.6	100.0
FastV	20%	69.4	57.0	**51.2**	59.2	**60.2**	**57.1**	54.2	**66.8**	59.3	96.3
VisionZip	20%	69.6	57.2	50.2	59.0	58.6	56.6	56.3	66.4	59.6	96.8
FastVID	20%	69.9	56.3	49.7	58.6	57.2	56.4	57.8	64.7	59.4	96.4
FlashVID	20%	**70.4**	**58.6**	49.7	**59.6**	59.2	56.8	**58.1**	66.5	**60.2**	**97.7**
FastV	10%	63.7	54.7	49.3	55.9	58.6	54.9	51.1	63.6	57.3	91.6
VisionZip	10%	67.0	54.7	47.6	56.4	55.8	55.5	54.5	64.3	57.7	93.7
FastVID	10%	66.3	53.6	49.0	56.3	56.0	55.6	55.4	62.3	57.4	93.2
FlashVID	10%	**68.1**	**54.7**	**49.0**	**57.3**	**57.4**	**55.9**	**57.1**	**65.5**	**58.9**	**95.6**

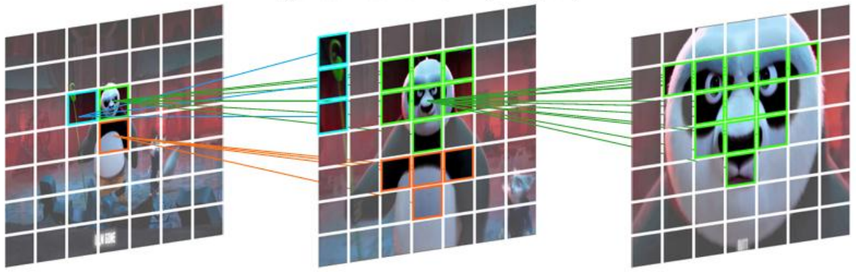
Comparison on Qwen2.5-VL

Method	Retention Ratio R	TFLOPs	Vision Encoding	Prefilling Time			TTFT	Avg.	
				Compression	LLM Forward	Total		Score	Rel. Acc (%)
Vanilla	100%	113.4	785.0	-	1220.8	1220.8 (1.0×)	2005.8 (1.0×)	58.9	100.0
FastVID	25%	22.4	785.0	28.6	273.2	301.8 (4.0×)	1086.8 (1.8×)	58.0	98.5
FlashVID	10%	8.6	785.0	60.2	133.1	193.3 (**6.3**×)	978.3 (**2.1**×)	58.4	**99.1**

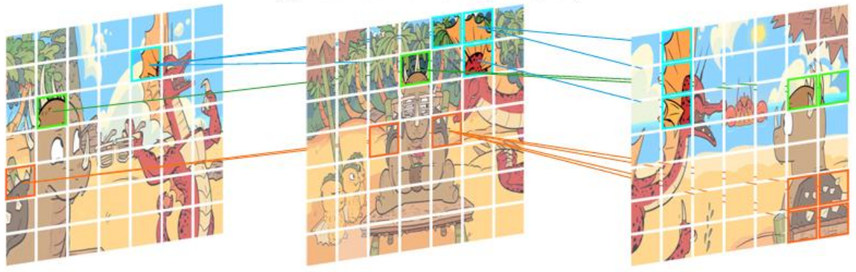
Efficiency of FlashVID on LLaVA-OneVision

Experiments

(a) Visualization of TSTM (Example 1)



(b) Visualization of TSTM (Example 2)



(c) Visualization of TSTM (Example 3)

Visualizations of TSTM

Question: *What kind of communication is listed before Semaphore?*

LLaVA-OneVision *C. Telegraph.* ✗

LLaVA-OneVision w/ FlashVID *D. Pony express.* ✓

(a) LLaVA-OneVision with and without FlashVID (Example 1)

Question: *What are the moves in the last scene of this dance?*

LLaVA-OneVision *B. Passe and then chasse* ✗

LLaVA-OneVision w/ FlashVID *A. Kneel down on one knee and lean back.* ✓

(b) LLaVA-OneVision with and without FlashVID (Example 2)

Question: *Who fight versus the black dinosaur at last?*

LLaVA-OneVision *C. A King Kong.* ✗

LLaVA-OneVision w/ FlashVID *D. A dragon* ✓

(c) LLaVA-OneVision with and without FlashVID (Example 3)

Question: *which smart phone is advertised on the screen of the laptop?*

LLaVA-OneVision *B. iPhone 14 Pro.* ✗

LLaVA-OneVision w/ FlashVID *D. iPhone 6s.* ✓

(d) LLaVA-OneVision with and without FlashVID (Example 4)

Qualitative Comparisons

1) HUANG, Yihong, et al. N\" uwa: Mending the Spatial Integrity Torn by VLM Token Pruning. *arXiv preprint arXiv:2602.02951*, 2026.

Nüwa: Mending the Spatial Integrity Torn by VLM Token Pruning[1)]

Introduction

- Motivation
  - 기존 VLM token pruning 방법론은 VQA task에서는 좋은 성능을 보이지만 VG task에서는 성능이 급락함
    - 해당 원인을 분석하기 위해 empirical analysis 진행
      - VQA는 이미지의 semantic 정보를 필요로 하지만 VG는 토큰 간의 spatial integrity가 필수적임
    - Simple baselines versus advanced pruning methods
      - 기존 pruning 방법론 VG task에서 큰 성능 저하를 보임
      - 오히려 average pooling이 더 좋은 성능을 보임
      - Pooling 방식이 spatial topology를 어느 정보 유지하기 때문

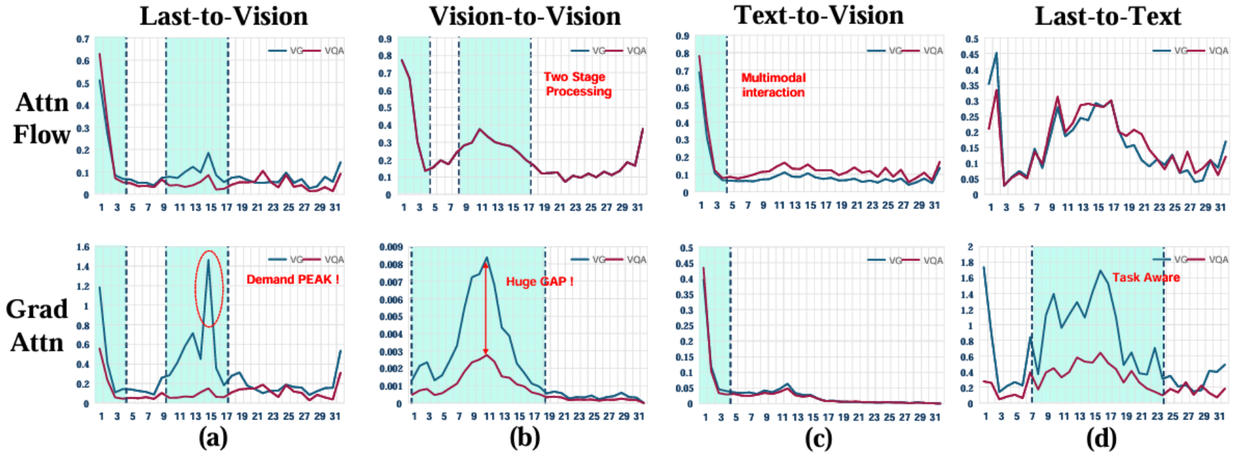
Avg Tokens	Method	Refcoco-test	Refcoco+-testA	Refcoco+-testB	Refcocog-test
576	LLaVA	58.30	59.43	38.88	48.50
128	FastV	10.34	8.53	9.83	8.87
	SparseVLM	6.27	5.79	4.22	6.35
	VisionZip	4.49	4.06	4.86	3.50
	Pooling	23.01	24.37	15.04	19.69
64	FastV	2.73	1.17	1.02	2.19
	SparseVLM	1.04	0.96	1.28	0.61
	VisionZip	4.04	3.73	3.86	3.38
	Pooling	12.01	12.20	7.55	11.40

Comparison on RefCOCO series, LLaVA1.5-7B

Finding 1 Advanced pruning methods provide limited benefits over simple baselines on VQA tasks, whereas all methods suffer systematic degradation on grounding tasks, with average pooling achieving the best performance.

Introduction

- Motivation
  - 기존 VLM token pruning 방법론은 VQA task에서는 좋은 성능을 보이지만 VG task에서는 성능이 급락함
    - Unveiling task-dependent visual processing pipeline
    - Attention flow와 gradient 분석
      - 모델이 answer를 decoding할 때 마지막 토큰이 시각 토큰들에 얼마나 집중하는지 시각화
      - 단순한 attention map뿐만 아니라 gradient-weighted attention을 사용해 실제 결과 예측에 기여하는 정보를 파악
      - 여기서 gradient는 역전파시 자동 계산되는 최종 출력 값에 대한 특정 layer의 attention map의 gradient 값
      - Mid layer에서 VG task의 경우 gradient-weighted attention이 상승하는 반면 VQA task에서는 비교적 낮은 의존도 기록



Attention Flow and Gradient Analysis on LLaVA1.5-7B

Introduction

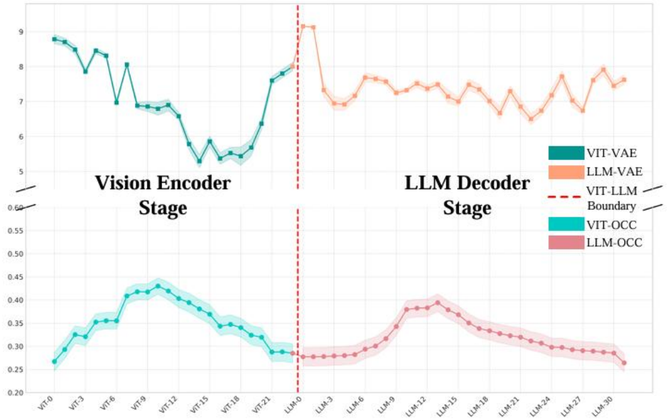
- Motivation
  - 기존 VLM token pruning 방법론은 VQA task에서는 좋은 성능을 보이지만 VG task에서는 성능이 급락함
    - Unveiling task-dependent visual processing pipeline
    - VAE와 OCC를 활용한 정량적 지표 분석

$$H(v_i) = -\sum_{j=1}^{i-1} p(v_j|v_i)\log_2 p(v_j|v_i), \quad \text{VAE} = \frac{1}{N-1}\sum_{i=2}^{N} H(v_i)$$

$$\text{OCC}(\mathcal{O}) = \frac{|V_k^{\text{model}} \cap V_{\mathcal{O}}|}{|V_k^{\text{model}} \cup V_{\mathcal{O}}|}$$

- Visual Attention Entropy (VAE)
  - ✓시각 attention이 global하게 퍼져 있는지 (높은 값) 혹은 local한 영역에 집중되어 있는지 (낮은 값) 수치화
  - ✓ViT 인코더의 layer가 깊어질수록 VAE가 감소, 즉 초반 globality를 고려하다 점차 fine-grained feature 추출로 전환
- Object-Centric Cohesion (OCC)
  - ✓토큰들이 실제 GT 객체와 얼마나 겹치는지를 IoU로 측정한 값
  - ✓ViT와 LLM 모두 mid layer에서 OCC 점수의 peak를 달성
  - ✓이는 모델이 중반부에서 object-level의 representation을 형성한다는 증거

Finding 2 Visual processing in VLMs unfolds through a **multi-stage pipeline**, progressing from global semantic integration to fine-grained object-centric focus, with **task-specific** reliance on vision tokens. Grounding tasks require heightened visual integration during middle stages for spatial reasoning, in contrast to the reduced demands in image understanding tasks.

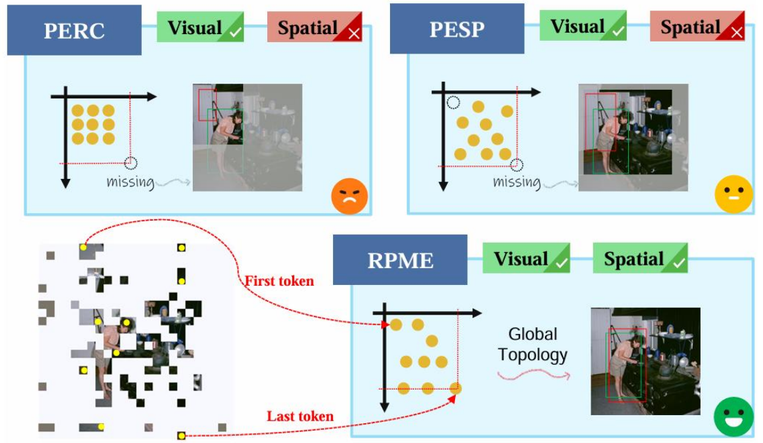


Layer-wise Analysis of VAE and OCC on LLaVA1.5-7B

1) YANG, Senqiao, et al. Visionzip: Longer is better but not necessary in vision language models. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025. p. 19792-19802.
2) CHEN, Liang, et al. 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*. Cham: Springer Nature Switzerland, 2024. p. 19-35.

Introduction

- Motivation
  - 기존 VLM token pruning 방법론은 VQA task에서는 좋은 성능을 보이지만 VG task에서는 성능이 급락함
    - Reconstructing the global reference frame
      - 기존 pruning 방법들이 왜 spatial 정보를 훼손하는지 positional embedding (PE) 처리 방식의 관점에서 분석
    - Pruning 시 PE의 3가지 유형
      - PERC (Position Embedding Range Compression) *e.g.*, VisionZip[1)]
        - ✓Pruning 후 남은 토큰들을 1부터 K까지 다시 mapping해 좁은 영역으로 압축
        - ✓기존 global 좌표계를 완전 상실
      - PESP (Position Embedding Sparse Preservation) *e.g.*, FastV[2)]
        - ✓남은 토큰들이 원래 가지고 있던 절대 위치 인덱스를 그대로 유지
        - ✓좌표 자체는 정확하지만 sparse한 상태가 되어 spatial continuity가 깨짐
      - RPME (Relative Position Mapping Extension) → proposed
        - ✓남은 토큰들의 상대적인 거리를 유지하면서 이를 전체 이미지 크기에 맞게 linear mapping해 확장
        - ✓개념 자체는 histogram eqalization이랑 유사하고 global frame을 복원해 모델의 이미지 전체 공간감을 복원



Different PE Strategies

Introduction

- Motivation
  - 기존 VLM token pruning 방법론은 VQA task에서는 좋은 성능을 보이지만 VG task에서는 성능이 급락함
    - Reconstructing the global reference frame
      - 대표적인 기존 방법론인 VisionZip (PERC 방식)과 FastV (PESP 방식)에 제안한 RPME를 적용해 성능 확인
      - 실험 결과 성능이 향상되는 것을 확인해 spatial continuity를 복원하는 것이 VG task에 도움이 됨을 검증

Method	Refcoco -test	Refcoco -val	Refcoco+ -testA	Refcoco+ -testB	Refcoco+ -val	Refcocog -test	Refcocog -val	GQA	MMB	VQAv2	MME
Vanilla	58.30	56.42	59.43	38.88	46.32	48.50	48.82	61.9	64.7	78.5	1862
Average 64 Tokens											
VisionZip-fix	11.57 (+7.53)	10.50 (+6.69)	9.27 (+5.54)	7.57 (+3.71)	8.62 (+5.12)	8.19 (+4.81)	8.31 (+5.10)	55.6 (+0.5)	61.8 (+1.7)	70.6 (-1.8)	1700 (+10)
Fastv-fix	4.52 (+1.79)	4.11 (+2.10)	3.84 (+2.67)	2.75 (+1.73)	4.31 (+1.90)	4.17 (+1.98)	4.22 (+2.21)	46.2 (+0.1)	47.8 (-0.2)	54.1 (-0.9)	1247 (-8)
Pooling	12.01	11.84	12.20	7.55	10.50	11.40	9.85	–	–	–	–
Average 128 Tokens											
VisionZip-fix	21.39 (+16.90)	21.04 (+16.93)	19.96 (+15.90)	13.45 (+8.59)	16.10 (+12.22)	15.69 (+12.19)	15.52 (+12.04)	58.5 (+0.9)	63.4 (+1.4)	74.3 (-1.3)	1751 (-10)
Fastv-fix	13.41 (+3.07)	13.24 (+3.11)	11.69 (+3.16)	12.29 (+2.46)	14.55 (+3.31)	12.02 (+3.15)	11.87 (+3.45)	51.3 (+0.8)	57.7 (+1.6)	60.3 (-1.5)	1494 (+4)
Pooling	23.01	22.67	24.37	15.04	17.88	19.69	19.03	–	–	–	–

Position Reconstruction Experiment Analysis on LLaVA1.5-7B

Finding 3 The degradation of VLMs on grounding tasks is principally driven by the loss of Global Spatial Reference Frame within token pruning strategies, which can be restored by preserving global position embedding.

Introduction

- Motivation
  - 기존 VLM token pruning 방법론은 VQA task에서는 좋은 성능을 보이지만 VG task에서는 성능이 급락함
    - Attention blocking experiment
      - LLaVA-1.5 7B 모델의 layer를 4단계로 나누고 특정 유형의 attention을 강제로 block했을 때 성능 변화 실험 진행
        - ✓Vision-to-vision: 시각 토큰끼리의 상호작용 차단
        - ✓Last-to-vision: 마지막 prediction 토큰이 시각 토큰을 보는 것을 차단
        - ✓Text-to-vision: 텍스트 토큰이 시각 토큰을 보는 것을 차단

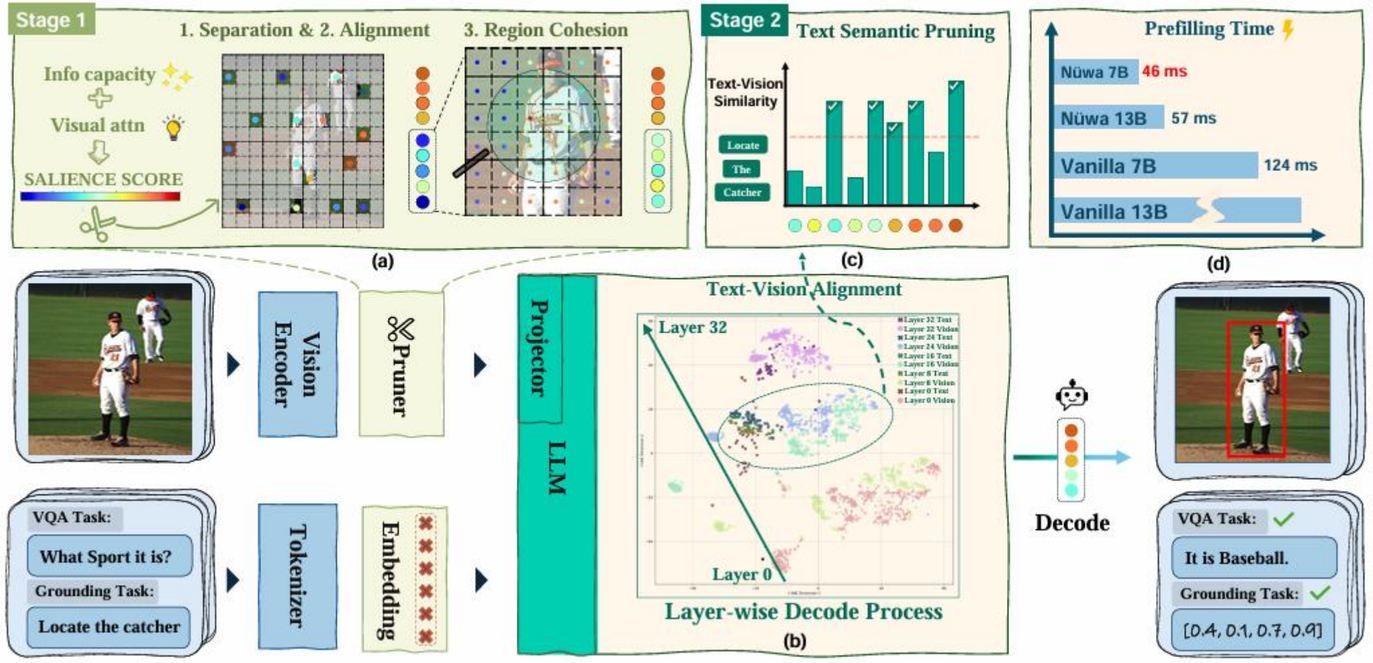
	Vision to Vision			Text to Vision			Last to Vision		
Blocked Layer	GQA	MMB	Refcoco-test	GQA	MMB	Refcoco-test	GQA	MMB	Refcoco-test
Original	61.9	64.7	58.30	61.9	64.7	58.30	61.9	64.7	58.30
0-7	60.5	61.62	18.17	39.38	51.63	54.75	62.55	64.26	57.85
8-15	55.2	57.98	**2.64**	53.32	46.39	14.11	62.47	64.08	**2.01**
16-23	58.44	62.37	19.27	61.04	62.45	15.94	61.32	64.26	64.52
24-31	58.81	62.37	19.31	61.12	63.05	15.08	62.12	64.17	59.10

Attention Blocking Experiments on LLaVA1.5-7B

Finding 4 Attention blocking experiments further reveal that grounding tasks primarily rely on two sets of information: 1. Abstract semantic information continuously extracted from text tokens to visual tokens; 2. Precise positional information was extracted from the model's mid-stage. Meanwhile, most VQA tasks predominantly depend on abstract semantic information extracted from text tokens to visual tokens during the model's early to mid-stages.

Methodology

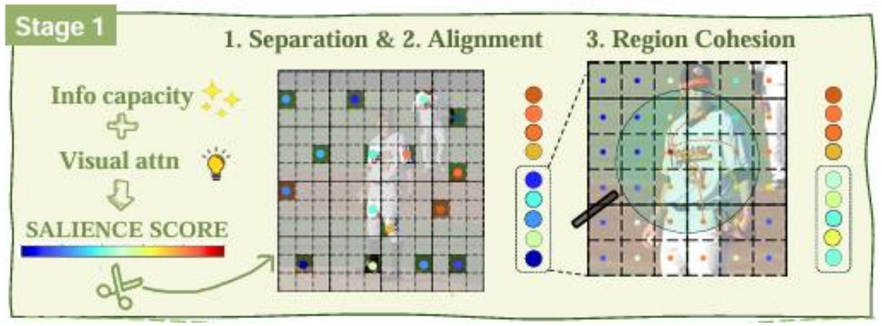
- Nüwa
  - Global spatial reference frame을 최대한 보존하는 pruning 기법 제안
    - 일관된 coverage를 보장하기 위한 spatial uniformity
    - Local salience를 유지하면서 vision-centric하고 cohesive하게 redundant information aggregation 수행
    - Textual semantics에 기반해 task-relevant 토큰을 선택하기 위한 text-modulated fine-grained filtering 적용



Nüwa Framework

Methodology

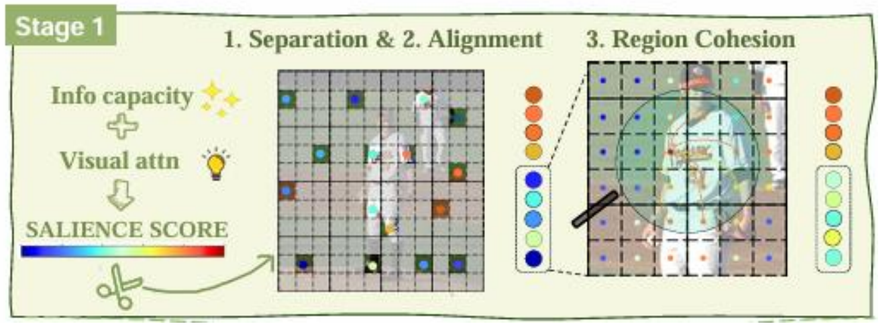
- Stage 1: Spatial Cohesion Pruning in the Vision Encoder
  - 공간 정보를 보존하면서 시각 토큰을 압축
  - Separation via grid partitioning
    - 입력 토큰 grid $\mathcal{T} = \{t_1, t_2, \dots, t_{N^2}\}$를 $M \times M$개의 non-overlapping local regions $\mathcal{R}_{i,j}$로 나눔
      - 전체 이미지를 한번에 처리하면 위치 정보가 섞이기 때문에 global coordinate system을 유지하기 위함
  - Alignment via salience identification
    - 각 영역 내에서 정보량이 가장 많은 benchmark tokens $\mathcal{T}_{\mathcal{B}}$ 선택
      - 단순히 CLS 토큰과의 attention score만 보면 깊은 layer에서 정보가 희소해지는 문제 존재
      - 따라서 information capacity를 나타내는 Key 벡터의 L2 Norm을 함께 고려한 salience score $S(t_i)$ 계산
      - $S(t_i) = \alpha_{cls,i} \cdot \left|\left|k_i\right|\right|_2$



Stage 1

Methodology

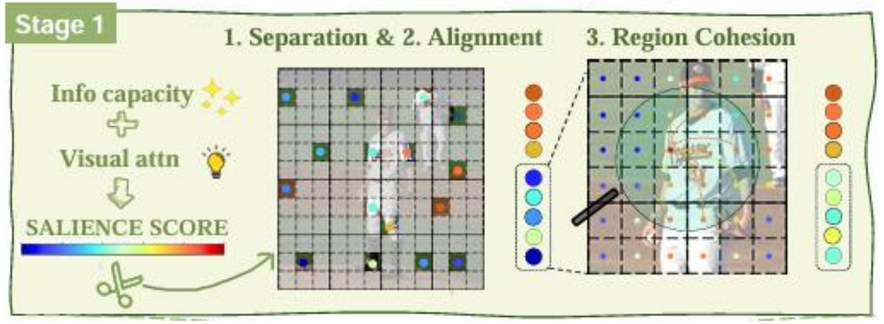
- Stage 1: Spatial Cohesion Pruning in the Vision Encoder
  - Aggregation via spatial proximity
    - Benchmark token들이 주변의 다른 토큰들의 정보를 흡수해 aggregate하는 과정
    - Benchmark token을 성격에 따라 두 그룹으로 나누어 role assignment 수행
      - Pillar tokens $\mathcal{T}_{\mathcal{P}}$: Benchmark Token 중 $||k_i||_2$ (L2-norm) 값이 상위 25%인 토큰
        - ✓Task-agnostic하게 절대적으로 중요한 정보를 담고 있을 확률이 높음 (Register 토큰처럼 동작)
        - ✓따라서 $\mathcal{T}_{\mathcal{P}}$는 자신의 값을 그대로 유지하고 다른 토큰의 정보를 섞지 않음
      - Collector tokens $\mathcal{T}_C$: 나머지 benchmark tokens
        - ✓주변 토큰들의 정보를 aggregate해 local 정보를 대표하는 역할 수행



Stage 1

Methodology

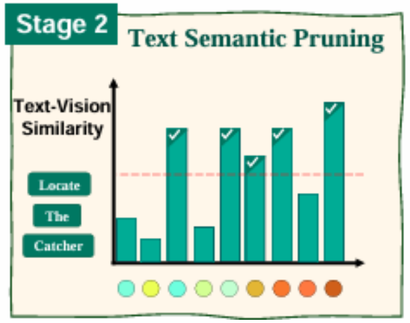
- Stage 1: Spatial Cohesion Pruning in the Vision Encoder
  - Aggregation via spatial proximity
    - $\mathcal{T}_C$는 semantic similarity와 spatial proximity를 동시에 고려해 weighted aggregation 형태로 주변 정보 흡수
      - Semantic similarity A_{ij}: 두 토큰 간의 cosine similarity
        - ✓ $A_{ij} = \mathrm{ReLU}\left(\mathrm{sim}\left(v_i, v_j\right)\right)$
      - Spatial proximity P_{ij}: 물리적 거리가 멀어지면 가중치를 낮춰 엉뚱한 위치의 정보가 섞이는 것 방지
        - ✓ $P_{ij} = 1 - \max\left(1, \frac{d(p_i, p_j)}{d_{thresh}}\right)$
      - 최종 aggregation weight W_{ij}와 토큰들의 특징 벡터를 곱해 업데이트된 benchmark token V_B' 형성
        - ✓ $W_{ij} = \begin{cases} \delta_{ij} & if\ t_i \in \mathcal{T}_P \\ A_{ij}P_{ij} & if\ t_i \in \mathcal{T}_C \end{cases}$



Stage 1

Methodology

- Stage 2: Text-modulated Pruning in the LLM
  - 질문과 관련 없는 시각 정보를 제거하는 것이 목적이며 LLM의 intermediate layer에서 수행
    - 앞선 empirical analysis와 같이 초반 제거 시 LLM 내부에서 multimodal interaction이 충분히 일어나지 않는다고 주장
  - Holistic textual query vector derivation
    - 텍스트 입력을 단순 average pooling해 하나의 대표 벡터 $\bar{q}$ 로 만듦
      - $\bar{q} = \frac{1}{K}\sum_{k=1}^{K} q_k$
  - Relevance scoring
    - 각 시각 토큰 v_i'이 $\bar{q}$와 얼마나 관련이 있는지 계산
      - 이때 시각 토큰은 projection layer를 거쳐 텍스트와 동일한 embedding space로 mapping된 상태
      - Relevance score R_i는 cosine similarity로 정의
  - Top-K selection
    - R_i가 높은 순서대로 상위 K_{final}개의 시각 토큰만 남김



Stage 2

Experiments

Method	Source	GQA	MMB	MMMU	MME	VQAv2	VQAtext	POPE	SQA	SEED	MMVet	avg
Vanilla	CVPR'24	61.9	64.7	36.3	1862	78.5	58.2	85.9	69.5	58.6	31.1	100%
Average Token 192 ↓ 66.7%												
FastV	ECCV'24	52.7	61.2	34.3	1612	67.1	52.5	64.8	67.3	57.1	27.7	89.53%
PDrop	CVPR'25	57.1	63.2	34.1	1766	74.9	56.1	82.3	**70.2**	54.7	30.5	95.87%
SparseVLM	ICML'25	57.6	62.5	33.8	1721	75.6	56.1	83.6	69.1	55.8	31.5	96.11%
VisionZip	CVPR'25	59.3	63.0	**36.6**	1782	**76.8**	57.3	85.3	68.9	56.4	**31.7**	98.26%
Nüwa	-	**60.9**	**64.3**	35.5	**1834**	75.9	**57.4**	**86.4**	68.2	**59.7**	30.5	**98.80%**
Average Token 128 ↓ 77.8%												
FastV	ECCV'24	49.6	56.1	34.9	1490	61.8	50.6	59.6	60.2	55.9	28.1	85.04%
PDrop	CVPR'25	56.0	61.1	34.2	1664	73.5	55.1	82.3	**69.9**	53.3	30.8	94.32%
SparseVLM	ICML'25	56.0	60.0	33.8	1696	73.8	54.9	80.5	67.1	53.4	30.0	93.36%
VisionZip	CVPR'25	57.6	62.0	**37.9**	1761	**75.6**	**56.8**	83.2	68.9	54.9	**32.6**	97.63%
PruMerge	ICCV'25	57.8	59.6	36.2	1712	74.7	54.3	81.5	67.6	-	30.4	95.06%
Nüwa	-	**60.2**	**63.4**	35.8	**1828**	75.1	**57.0**	**85.5**	67.8	**58.7**	29.8	**97.87%**
Average Token 64 ↓ 88.9%												
FastV	ECCV'24	46.1	48.0	34.0	1256	55.0	47.8	59.6	51.1	51.9	25.8	79.36%
PDrop	CVPR'25	41.9	33.3	26.5	1092	57.3	45.9	55.9	69.2	40.0	24.9	71.56%
SparseVLM	ICML'25	53.8	60.1	35.44	1589	68.2	53.4	77.5	**69.8**	51.1	24.9	89.93%
VisionZip	CVPR'25	55.1	60.1	36.2	1690	72.4	**55.5**	77.0	69.0	52.2	**31.7**	93.99%
PruMerge	ICCV'25	55.4	59.6	35.8	1616	71.3	52.0	75.7	69.5	-	28.0	91.71%
Nüwa	-	**58.3**	**62.0**	**36.4**	**1706**	**72.8**	54.9	**83.0**	67.5	**56.44**	28.2	**94.91%**

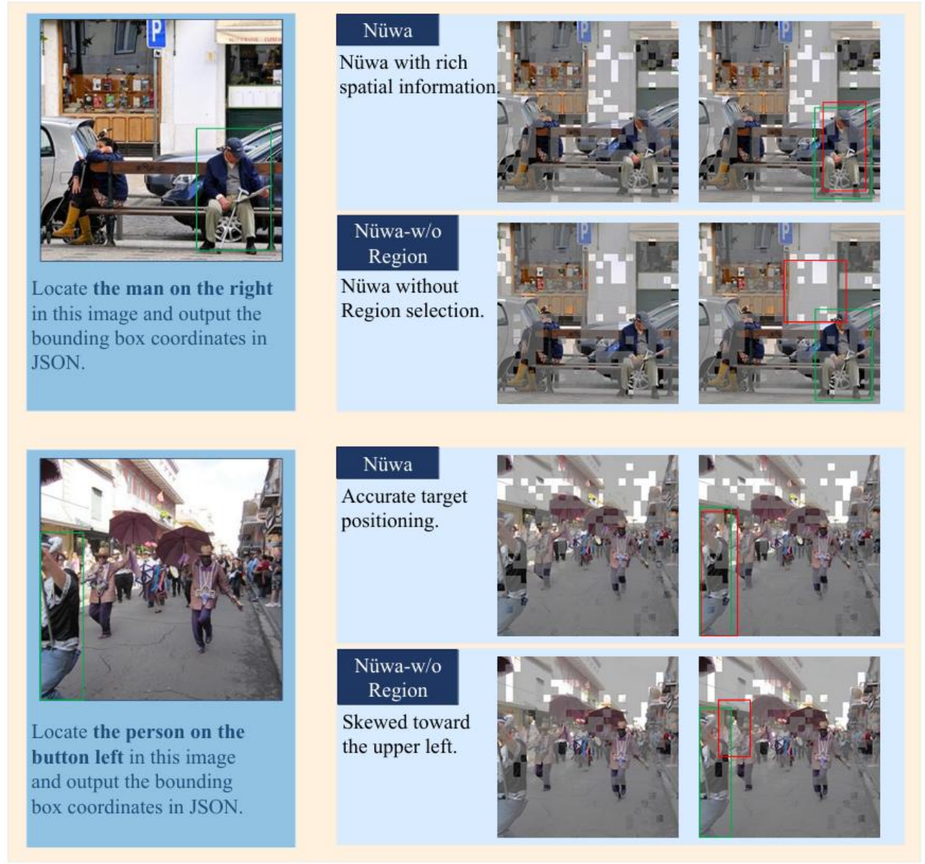
VQA Performance Comparison on LLaVA-1.5 7B

Method	Source	Refcoco-test	Refcoco-val	Refcoco+-testA	Refcoco+-testB	Refcoco+-val	Refcocog-test	Refcocog-val	avg
Vanilla	CVPR'24	58.30	56.42	59.43	38.88	46.32	48.50	48.82	100%
Average Tokens 192 ↓ 66.7%									
FEATHER*	ICCV'25	27.7	-	24.7	-	-	27.2	-	48.38%
Nüwa	-	**47.91**	**46.12**	**43.18**	**31.86**	**37.68**	**37.64**	**37.90**	**79.29%**
Average Tokens 128 ↓ 77.8%									
Fastv	ECCV'24	10.34	10.13	8.53	9.83	8.16	8.87	9.10	18.55%
SparseVLM	ICML'25	6.27	6.17	5.79	4.22	9.85	6.35	6.47	12.84%
VisionZip	CVPR'25	4.49	4.11	4.06	4.86	3.88	3.50	3.48	8.1%
Nüwa	-	**45.09**	**43.69**	**42.63**	**28.98**	**35.32**	**36.59**	**36.00**	**75.20%**
Average Tokens 64 ↓ 88.9%									
Fastv	ECCV'24	2.73	2.01	1.17	1.02	2.41	2.19	2.01	3.81%
SparseVLM	ICML'25	1.04	1.01	0.96	1.28	0.96	0.61	0.66	1.88%
VisionZip	CVPR'25	4.04	3.81	3.73	3.86	3.50	3.38	3.21	7.28%
Nüwa	-	**29.43**	**28.60**	**28.22**	**17.47**	**22.22**	**21.81**	**21.42**	**47.19%**

VG Performance Comparison on LLaVA-1.5 7B

Method	Avg Token	main (TFLOPs)	metric (MFLOPs)	Prefill-Time (ms)
Vanilla	576	5.9730	0	124
FastV	64	0.8341	**4.7185**	92 ↓ 26%
SparseVLM	64	0.8141	5.5050	104 ↓ 16%
VisionZip	64	**0.6461**	8.9128	**45** ↓ 63%
Nüwa	64	0.6476	17.5636	46 ↓ 62%

Efficiency Analysis



Comparison of Regional Partitioning

Thank you