2026 동계 세미나

Efficient Token Pruning for Large Vision-Language Models

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
변재현

Outline

- Background
  - What is Token Pruning?
  - LVLM Token Pruning

- Beyond Attention or Similarity: Maximizing Conditional Diversity for Token Pruning in MLLMs
  - NeurIPS 2025

- SparseVILA: Decoupling Visual Sparsity for Efficient VLM Inference
  - ICCV 2025

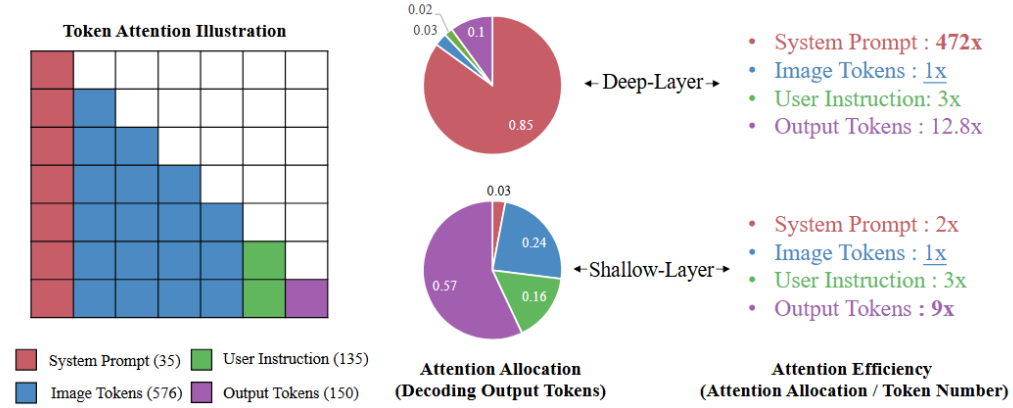
Background

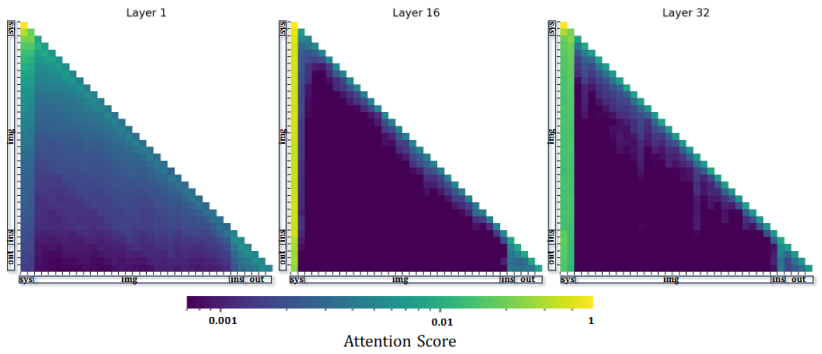
- What is Token Pruning?
  - Token Pruning의 개념
    - Transformer 모델의 입력 토큰 중 “덜 중요한” 토큰을 선별하여 제거하는 기법
    - Attention 연산의 complexity ($O(N^2)$) 문제를 완화하여 추론 속도 향상
  - 작동 원리
    - Ranking
      - Importace 기반 점수 산출 (e.g., attention score)
    - Filtering
      - 상위 K개 토큰 유지 또는 threshold 기반 필터링
    - Inference
      - 줄어든 토큰으로 연산 수행
  - 가장 핵심은 입력 토큰은 줄여서 추론 속도는 향상되지만, 성능은 유지

1) Chen, Liang, et al. "An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024.

Background

- LVLM Token Pruning
  - LVLM의 computational bottleneck
    - 이미지 해상도가 높을수록 visual token 수가 급격히 증가
    - Visual token이 입력의 대부분을 차지하지만, text 생성 시 attention 비중이 낮음
  - FastV[1] 등장
    - LVLM의 입력의 대부분은 visual token이지만 모델이 깊어질수록 visual token에 대한 attention이 급격히 떨어짐
      - ∴ 후반 layer에서는 불필요한 visual token을 연산하느라 자원을 낭비
    - K번째 layer까지는 정상적으로 연산하되, 이후 중요도가 R% 이하의 visual token은 제거





Beyond Attention or Similarity: Maximizing Conditional Diversity for Token Pruning in MLLMs

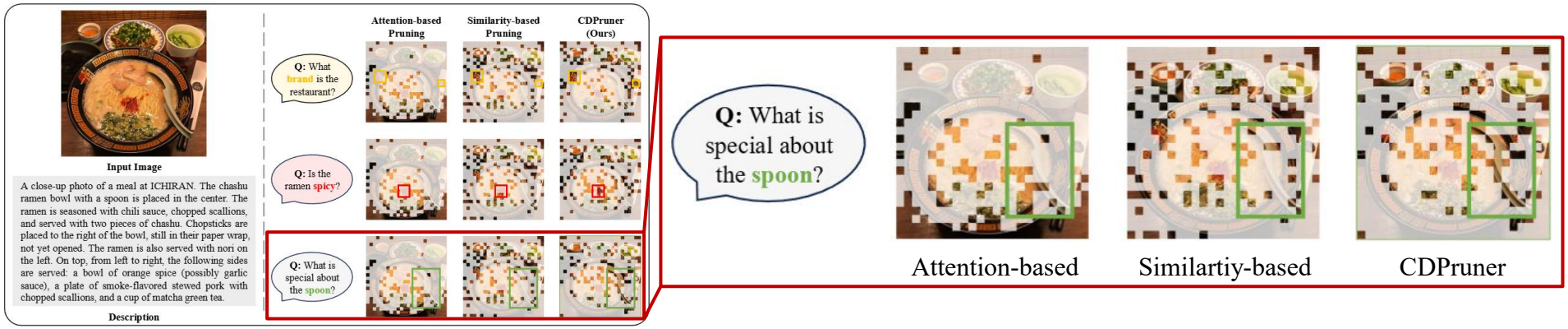
[NeurIPS 2025]

CDPruner[2)]

- Introduction
  - 기존 pruning 방식의 한계
    - Attention-based (e.g., FastV)
      - Attention score가 높은 토큰만 유지하는 경우, 시각적으로 유사한 토큰들이 많이 남는 경향이 발생 (Redundancy 문제)
    - Similarity-based
      - Feature가 유사한 토큰을 제거할 경우, 사용자의 instruction과 관련된 중요한 토큰까지 제거될 위험 존재 (Context 무시)
  - Motivation
    - 단순히 “중요한” 토큰이 아니라, 다양하면서도 질문과 관련된 토큰을 남겨야 함
    - Conditional Diversity 도입

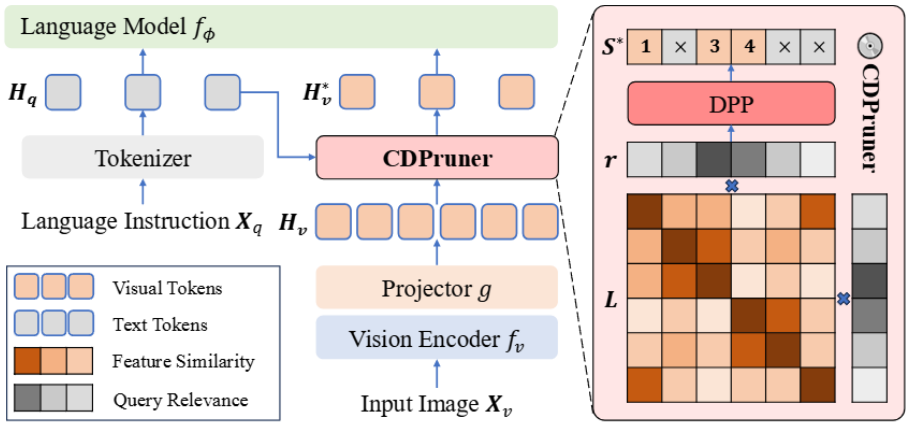
CDPruner[2)]

- Introduction
  - CDPruner (Conditional Diversity Pruner)
    - 핵심 개념
      - Instruction(텍스트 질문)을 조건부로 하여 visual token의 다양성을 최대화
      - DPP(Determinantal Point Process)를 활용한 최적화 문제로 재정의
    - 특징
      - Training-free: 별도의 학습 과정 불필요
      - Model-agnostic: 다양한 MLLM 아키텍처에 즉시 적용 가능



CDPruner[2)]

- Method
  - CDPruner pipeline
    1. Input: Visual tokens, Text instruction
    2. Visual similarity calculation: 시각 토큰 간의 cosine similarity 행렬 계산
    3. Instruction relevance estimation: Text와 visual token 간의 attention map을 분석하여 각 visual token의 중요도 산출
    4. 2번과 3번을 결합하여 text에 조건화된 새로운 유사도 행렬 생성
    5. Subset selection: DPP(Determinantal Point Process)를 통해 조건부 다양성을 최대화하는 토큰 부분집합 선택
    6. Pruning: 선택되지 않은 토큰 제거 후 MLLM 입력

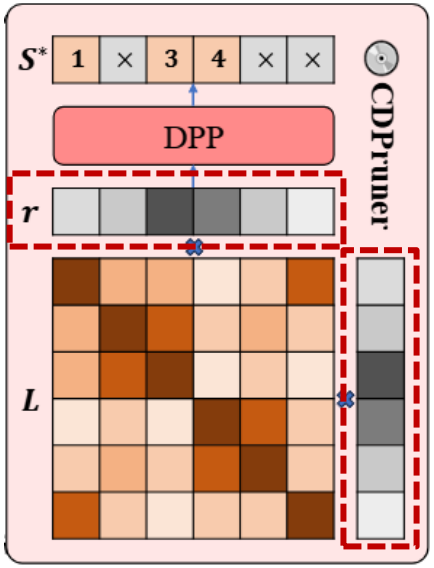


CDPruner[2)]

- Method
  - DPP (Determinantal Point Process)
    - 양자 물리학에서 "Fermion"이라는 입자들이 서로 같은 상태에 있지 않으려는 성질을 모델링하기 위해 만들어짐
      - 중복을 피하고, 다양성을 확보
    - 유사도 행렬 L
      - 모든 visual token 쌍 (i, j)에 대해 cosine similarity를 계산
      - L_{ij}가 1에 가까우면 해당 쌍이 유사, 0에 가까우면 서로 다른 특징을 가짐
    - 부분집합 S가 선택될 확률
      - 어떤 부분집합 S(visual tokens)가 선택될 확률 P(S)는 det(S)에 비례
        - ✓Vector들이 비슷할수록 det는 0에 가까워지고, 서로 직교할수록 det 커짐
      - S* = argmax det(L_S)
        - ✓가능한 모든 조합들 중, 행렬식이 가장 커서 최종 선택된 조합 S*

CDPruner[2)]

- Method
  - Insturction relevance
    - Visual token만 고려하면 이미지적으로만 다양한 token들만 남게 됨
      - 질문에 따라 남겨야할 중요한 정보가 다름
    - 질문 text 관련성 점수(r_i)
      - $r_i = \frac{H_v^i \cdot \bar{H}_q}{\|H_v^i\| \cdot \|\bar{H}_q\|}$ (Visual token: H_v^i, Text embedding: $\bar{H}_q$)
    - Normalization
      - $\tilde{r} = \frac{r - \min(r)}{\max(r) - \min(r)}$



CDPruner[2)]

- Method
  - CDPruner
    - 조건부 커널 행렬 생성
      - 기존 유사도 행렬 L에 질문 연관성 점수 $\tilde{r}$을 반영하여 새로운 행렬 $\tilde{L}$ 생성
        - ✓$\tilde{L}_{ij} = r_i \cdot L_{ij} \cdot r_j$
      - $\tilde{L} = \mathrm{diag}(\tilde{r}) \cdot L \cdot \mathrm{diag}(\tilde{r})$
    - $\tilde{L}$을 기반으로 부분집합 S의 점수 계산
      - $\log\det(\tilde{S}) = \sum_{i \in S} \log\left(\widetilde{r_i^2}\right) + \log\det(S)$
    - 최적의 조합 찾기
      - 점수가 가장 높은 조합을 찾아야하는데 모든 조합 계산은 비용 과다
      - Greedy algorithm을 사용
        - ✓수학적으로 최적해의 63% 이상의 성능을 보장
        - ✓Cholesky decomposition을 사용하여, $O(nm^2)$까지 줄임

CDPruner[2)]

- Method
  - CDPruner (Cholesky decomposition)

$L = V \cdot V^T$ cholesky decomposition.

$$L = \begin{bmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \rightarrow V = \begin{bmatrix} V_{11} & 0 & 0 \\ V_{21} & V_{22} & 0 \\ V_{31} & V_{32} & V_{33} \end{bmatrix}$$

- $V_{11}^2 = L_{11} \quad \therefore V_{11} = \sqrt{L_{11}}$
- $V_{21} \cdot V_{11} = L_{21} \quad \therefore V_{21} = \frac{L_{21}}{V_{11}}$
- $V_{31} \cdot V_{11} = L_{31} \quad \therefore V_{31} = \frac{L_{31}}{V_{11}}$
- $V_{21}^2 + V_{22}^2 = L_{22} \quad \therefore V_{22} = \sqrt{L_{22} - V_{21}^2}$
- $V_{32} \cdot V_{22} + V_{31} \cdot V_{21} = L_{32} \quad \therefore V_{32} = \frac{L_{32} - V_{31} \cdot V_{21}}{V_{22}}$

⋮

$$V = \begin{bmatrix} \sqrt{L_{11}} & 0 & 0 \\ \frac{L_{21}}{V_{11}} & \sqrt{L_{22} - V_{21}^2} & 0 \\ \frac{L_{31}}{V_{11}} & \frac{L_{32} - V_{31} \cdot V_{21}}{V_{22}} & \sqrt{L_{33} - V_{31}^2 - V_{32}^2} \end{bmatrix}$$

- 대각선 (V_{ii}) : $\sqrt{\text{원래값} - (\text{앞 열 성분들의 제곱합})}$
- 나머지 (V_{ji}) : $\frac{\text{원래값} - (\text{앞 열 성분들의 내적 합})}{\text{현재 열의 대각선 값 } (V_{ii})}$

⇒ 행렬식 (det)는 Cholesky 분해된 대각선 성분들의 제곱의 곱.

$$\det(L) = V_{11}^2 \times V_{22}^2 \times V_{33}^2 \cdots$$

기존방법 : 남은 토큰을 하나씩 넣으면서 det 계산.

→ 해당 토큰을 넣으면 전체 부피 (det)가 얼마나 커짐?

Cholesky : 앞의 V_{11}^2, V_{22}^2은 계산했으니 새로운 V_{33}^2이 가장 큰 토큰을 선택.

→ 해당 토큰을 넣으면 대각선 성분 (V_{kk})가 얼마나 커짐?

→ 하나의 토큰을 뽑을 때마다, 후보 토큰들에서 뽑은 토큰과 겹치는 성분을 빼고 갱신함.

"최종적으로 이번에 뽑은 성분이 가장 큰 토큰 후보 선택"

CDPruner[2)]

- Experiments
  - LLaVA-1.5-7B

Method	VQA^{V2}	GQA	VizWiz	SQA^{IMG}	VQA^{Text}	POPE	MME	MMB^{EN}	MMB^{CN}	MMVet	Acc.	Rel.
Upper Bound, All 576 Tokens (100%)												
LLaVA-1.5-7B	78.5	61.9	50.1	69.5	58.2	85.9	1506.5	64.7	58.1	31.3	63.4	100.0%
Retain 128 Tokens (↓ 77.8%)												
FastV (ECCV24)	71.0	54.0	51.9	69.2	56.4	68.2	1368.9	63.0	55.9	27.0	58.5	92.8%
PDrop (CVPR25)	74.3	57.1	49.4	70.1	56.7	77.5	1444.1	62.3	55.3	27.6	60.3	95.0%
SparseVLM (ICML25)	75.1	57.3	49.7	69.0	56.3	83.1	1399.3	62.6	56.9	29.7	61.0	96.3%
PruMerge+ (2024.05)	75.0	58.2	53.7	69.1	54.0	83.1	1408.1	61.8	55.8	30.4	61.2	96.8%
TRIM (COLING25)	75.4	58.4	51.6	68.6	52.2	85.3	1413.4	63.0	52.3	29.9	60.7	95.8%
VisionZip (CVPR25)	75.6	57.6	51.6	68.7	56.9	83.3	1436.9	62.1	57.0	31.6	61.6	97.6%
DART (2025.02)	74.7	57.9	52.8	69.1	56.3	80.4	1408.7	60.7	57.3	30.9	61.1	96.9%
DivPrune (CVPR25)	76.0	59.4	52.8	68.6	55.9	87.0	1405.1	61.5	54.8	30.6	61.7	97.5%
CDPruner (Ours)	76.6	59.9	52.8	69.0	56.2	87.7	1431.4	63.1	55.0	32.8	**62.5**	**99.0%**
Retain 64 Tokens (↓ 88.9%)												
FastV (ECCV24)	55.9	46.0	49.1	70.1	51.6	35.5	973.5	50.1	42.1	18.9	46.8	74.9%
PDrop (CVPR25)	56.3	46.1	46.3	68.8	49.2	40.8	982.2	48.0	36.6	17.7	45.9	72.9%
SparseVLM (ICML25)	66.9	52.0	49.4	69.2	52.1	69.7	1190.4	58.3	49.6	24.4	55.1	87.1%
PruMerge+ (2024.05)	71.3	55.4	53.7	69.5	52.0	75.7	1316.8	59.6	52.1	28.0	58.3	92.4%
TRIM (COLING25)	72.4	56.6	51.1	69.0	49.7	85.9	1350.9	60.9	48.2	24.8	58.6	91.6%
VisionZip (CVPR25)	72.4	55.1	52.9	69.0	55.5	77.0	1365.2	60.1	55.4	29.4	59.5	94.4%
DART (2025.02)	71.3	54.7	53.5	69.3	54.7	73.8	1365.1	59.5	54.0	26.5	58.6	92.6%
DivPrune (CVPR25)	74.1	57.5	53.6	68.0	54.5	85.5	1334.7	60.1	52.3	28.1	60.0	94.7%
CDPruner (Ours)	75.4	58.6	53.4	68.1	55.3	87.5	1415.1	61.1	53.2	30.5	**61.4**	**97.0%**
Retain 32 Tokens (↓ 94.4%)												
PruMerge+ (2024.05)	65.6	52.9	53.5	67.9	49.2	66.7	1236.6	55.1	45.9	24.7	54.3	86.1%
TRIM (COLING25)	68.6	54.5	50.7	68.1	47.6	84.9	1251.8	57.7	40.1	20.5	55.5	86.2%
VisionZip (CVPR25)	67.1	51.8	52.4	69.1	53.1	69.4	1251.2	57.0	50.3	25.3	55.8	88.4%
DART (2025.02)	67.1	52.9	52.5	69.3	52.2	69.1	1273.3	58.5	50.0	25.0	56.0	88.6%
DivPrune (CVPR25)	71.2	54.9	53.3	68.6	52.9	81.5	1284.9	57.6	49.1	26.3	58.0	91.3%
CDPruner (Ours)	73.6	57.0	53.1	69.5	53.2	87.9	1373.0	59.6	49.6	27.8	**60.0**	**94.3%**

CDPruner[2)]

- Experiments
  - LLaVA-NeXT-7B (High resolution)

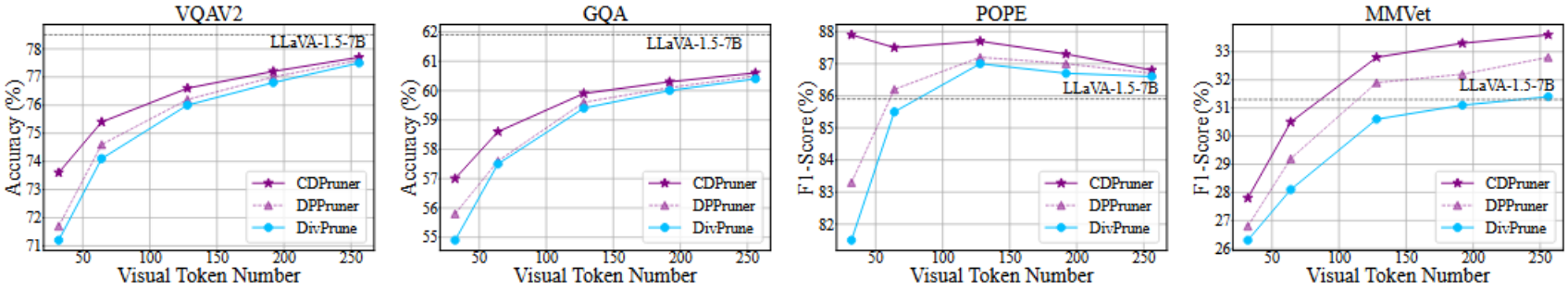
Method	VQA^{V2}	GQA	VizWiz	SQA^{IMG}	VQA^{Text}	POPE	MME	MMB^{EN}	MMB^{CN}	MMVet	Acc.	Rel.
Upper Bound, All 2880 Tokens (100%)												
LLaVA-NeXT-7B	81.3	62.5	55.2	67.5	60.3	86.8	1511.8	65.8	57.3	40.0	65.2	100.0%
Retain 640 Tokens (↓ 77.8%)												
FastV (ECCV24)	77.0	58.9	53.9	67.4	58.1	79.5	1412.6	63.1	53.5	39.5	62.2	95.6%
PDrop (CVPR25)	79.1	60.0	53.8	66.7	57.8	83.8	1475.9	64.1	55.2	36.7	63.1	96.5%
SparseVLM (ICML25)	79.2	61.2	53.6	67.6	59.7	85.3	1456.8	65.9	58.6	36.1	64.0	97.9%
PruMerge+ (2024.05)	78.2	60.8	57.9	67.8	54.9	85.3	1480.2	64.6	57.3	32.7	63.4	96.6%
TRIM (COLING25)	78.3	62.1	54.8	66.9	54.8	86.9	1471.8	66.8	55.8	37.8	63.8	97.6%
VisionZip (CVPR25)	79.1	61.2	57.1	68.1	59.9	86.0	1493.4	65.8	58.1	38.9	64.9	99.5%
DART (2025.02)	78.3	61.3	57.0	68.2	59.5	85.0	1450.2	64.9	57.1	36.9	64.1	98.2%
DivPrune (CVPR25)	79.3	61.9	55.7	67.8	57.0	86.9	1469.7	65.8	57.3	38.0	64.3	98.5%
CDPruner (Ours)	79.9	62.6	55.6	67.9	58.4	87.3	1474.5	66.3	57.5	41.9	**65.1**	**100.1%**
Retain 320 Tokens (↓ 88.9%)												
FastV (ECCV24)	61.5	49.8	51.3	66.6	52.2	49.5	1099.0	53.4	42.5	20.0	50.2	76.9%
PDrop (CVPR25)	66.8	50.4	49.7	66.7	49.0	60.8	1171.5	55.5	44.7	24.0	52.6	80.3%
SparseVLM (ICML25)	74.6	57.9	54.2	67.2	56.5	76.9	1386.1	63.1	56.7	32.8	60.9	93.3%
PruMerge+ (2024.05)	75.3	58.8	57.7	68.1	54.0	79.5	1444.3	63.0	55.6	31.4	61.6	94.0%
TRIM (COLING25)	74.9	59.9	53.5	66.2	50.2	86.5	1443.8	63.5	51.0	32.7	61.1	92.9%
VisionZip (CVPR25)	76.2	58.9	56.2	67.5	58.8	82.3	1397.1	63.3	55.6	35.8	62.4	95.7%
DART (2025.02)	75.7	59.5	56.8	67.5	57.6	81.0	1419.5	64.2	55.7	35.7	62.5	95.8%
DivPrune (CVPR25)	77.2	61.1	55.6	67.7	56.2	84.7	1423.3	63.9	55.7	34.8	62.8	96.0%
CDPruner (Ours)	78.4	61.6	55.8	67.8	57.4	87.2	1453.0	65.5	55.7	37.9	**64.0**	**98.0%**
Retain 160 Tokens (↓ 94.4%)												
PruMerge+ (2024.05)	70.5	56.2	57.2	66.9	50.3	71.1	1289.6	58.0	48.9	29.3	57.3	87.7%
TRIM (COLING25)	71.0	57.4	52.9	65.5	45.8	84.8	1275.8	61.6	45.2	29.6	57.8	87.7%
VisionZip (CVPR25)	71.4	55.2	55.5	67.9	55.0	74.9	1327.8	58.6	50.4	32.3	58.8	90.0%
DART (2025.02)	72.5	56.8	56.7	67.8	54.9	75.3	1325.4	62.0	53.6	32.2	59.8	91.7%
DivPrune (CVPR25)	75.0	59.3	56.1	67.1	54.1	80.0	1356.6	62.9	53.7	32.0	60.8	92.9%
CDPruner (Ours)	76.7	60.8	55.2	67.5	55.4	86.8	1425.3	64.2	53.8	36.2	**62.8**	**96.0%**

CDPruner[2)]

- Experiments
  - Efficiency analysis of different methods on LLaVA-NeXT-7B

Method	# Token	FLOPs (T)	Prefill Time (ms/token)	Decode Time (ms/token)	KV Cache (MB)	GPU Memory (GB)	Score (F1)
LLaVA-NeXT-7B	2880	41.7	246	29	1440.0	16.7	86.8
FastV (ECCV24)	320	4.4 (×9.5)	54 (×4.6)	23 (×1.2)	160.3	15.6	49.5
PDrop (CVPR25)	320	4.5 (×9.3)	55 (×4.5)	24 (×1.2)	160.2	15.6	60.8
SparseVLM (ICML25)	320	4.5 (×9.3)	71 (×3.5)	25 (×1.1)	161.2	18.6	76.9
VisionZip (CVPR25)	320	**4.2 (×9.9)**	**38 (×6.6)**	**22 (×1.3)**	**160.0**	14.8	82.3
DivPrune (CVPR25)	320	**4.2 (×9.9)**	**38 (×6.6)**	**22 (×1.3)**	**160.0**	**13.8**	84.7
CDPruner (Ours)	320	**4.2 (×9.9)**	**38 (×6.6)**	**22 (×1.3)**	**160.0**	**13.8**	**87.2**

  - Ablation study



CDPruner[2)]

- Conclusion
  - 기존 한계 극복
    - Attention 기반의 중복성 문제와 Similarity 기반의 문맥 무시 문제를 동시에 해결하여 정보 손실 최소화
  - CDPruner 제안
    - 사용자 질문(instruction)과의 연관성을 반영한 '조건부 다양성(Conditional Diversity)' 개념 도입
  - 수학적 최적화
    - DPP(행렬식 점 과정)를 활용해 정보량과 연관성을 동시에 최대화하는 토큰 부분집합 선별
  - 높은 범용성
    - 별도의 재학습이 필요 없는 Training-free 방식이며, 다양한 MLLM 아키텍처에 즉시 적용 가능

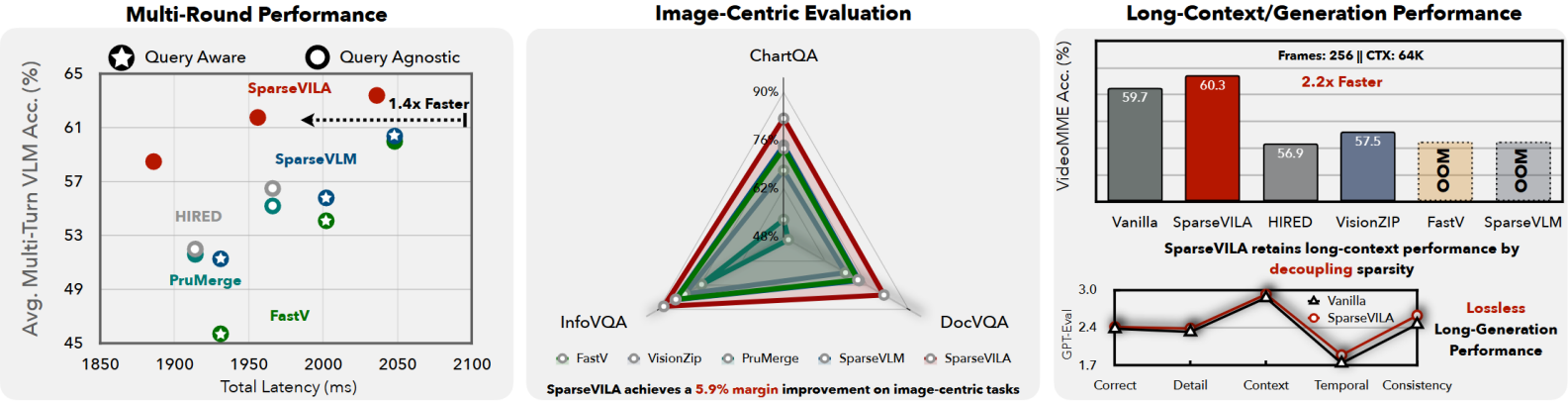
SparseVILA: Decoupling Visual Sparsity for Efficient VLM Inference [ICCV 2025]

SparseVILA[3)]

- Introduction
  - VLM Inference의 두 단계
    - Prefill Stage (Context Encoding)
      - 이미지와 텍스트 입력을 처리하여 KV Cache 생성 (Compute-bound)
    - Decoding Stage (Generation)
      - 토큰을 하나씩 생성 (Memory-bound)
  - 기존 한계
    - Visual token이 너무 많아 prefill 단계의 연산량 과다
    - Decoding 단계에서 긴 context로 인해 KV cache 메모리 부족 및 속도 저하
    - 기존 pruning은 한 번 제거하면 복구 불가능
      - Multi-turn 대화에서 이전 정보를 잃어버림

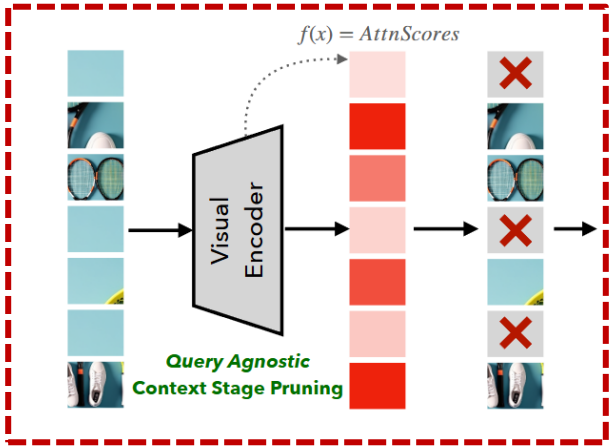
SparseVILA[3)]

- Introduction
  - SparseVILA's Decoupling
    - Prefill 단계와 decoding 단계의 sparsity 전략을 분리해야 함
      - Prefill: "압축" 관점
      - Decoding: "검색" 관점 (필요한 정보만 인출)
  - 목표
    - 초기 처리 속도 향상
    - Multi-turn 대화 능력 유지
    - 긴 비디오와 텍스트 처리 가능



SparseVILA[3)]

- Method
  - Prefill stage
    - 사용자가 준 정보를 읽고 이해하는 단계
    - Query-agnostic pruning
      - Vision encoder에서 나온 attention score map 기반 pruning
      - Text(query)가 들어오기 전, 이미지 encoding 단계에서 시각적 중복 제거
      - Mean-column sum을 계산하여 토큰의 중요도 측정
        - ✓다른 토큰들이 해당 토큰을 얼마나 attend하는 합산하는 방식
    - Visual Saliency S_j
      - $S_j = \frac{1}{H}\sum_{h=1}^{H}\sum_{i=1}^{N} A_{i,j}^{(h)}$
    - Pruning
      - S_j 기준 하위 토큰 제거

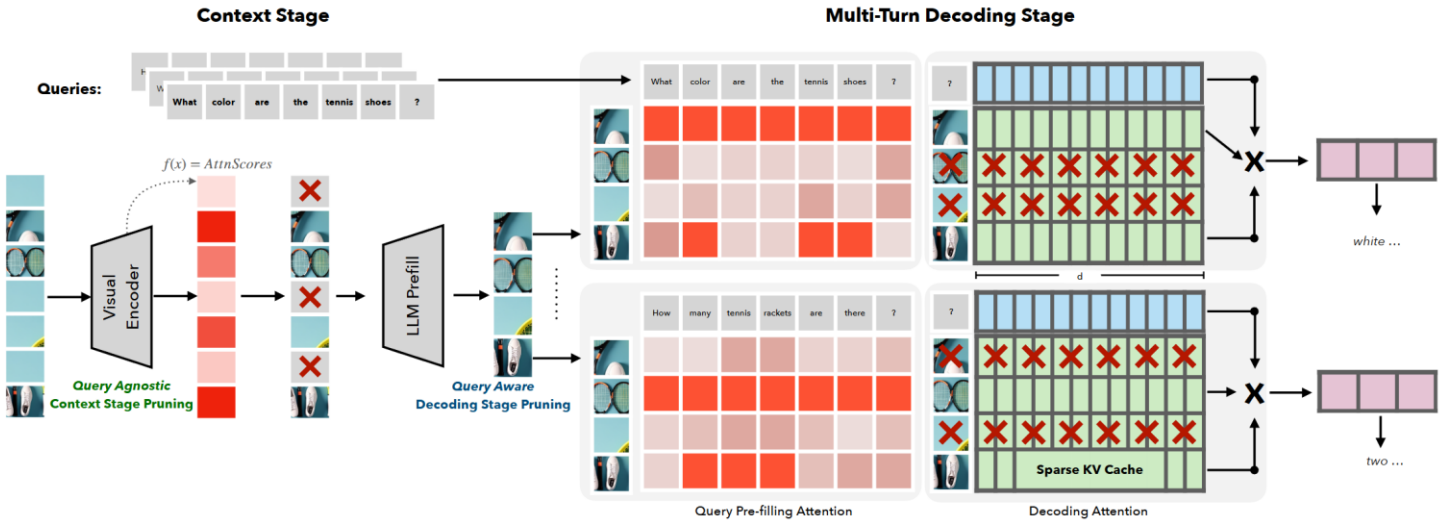


SparseVILA[3)]

- Method
  - Decoding stage
    - 실제로 답변을 출력하는 단계
    - Query-aware pruning
      - 생성 단계(decoding)에서 현재 질문에 필요한 토큰만 동적으로 retrieval
      - Query(text)와 key 간의 attention score를 계산하여 선정
    - Query-aware saliency(S_j^{dec})
      - $S_j^{dec} = \frac{1}{H \times L_Q} \sum_{h=1}^{H} \sum_{t=1}^{L_Q} \text{Weights}_{t,j}^{(h)}$
      - 모든 head와 모든 질문 토큰들이 시각 토큰 j를 얼마나 필요한가를 합산
      - Saliency가 높은 상위 K% 토큰만 선택하여 실제 attention 연산 수행

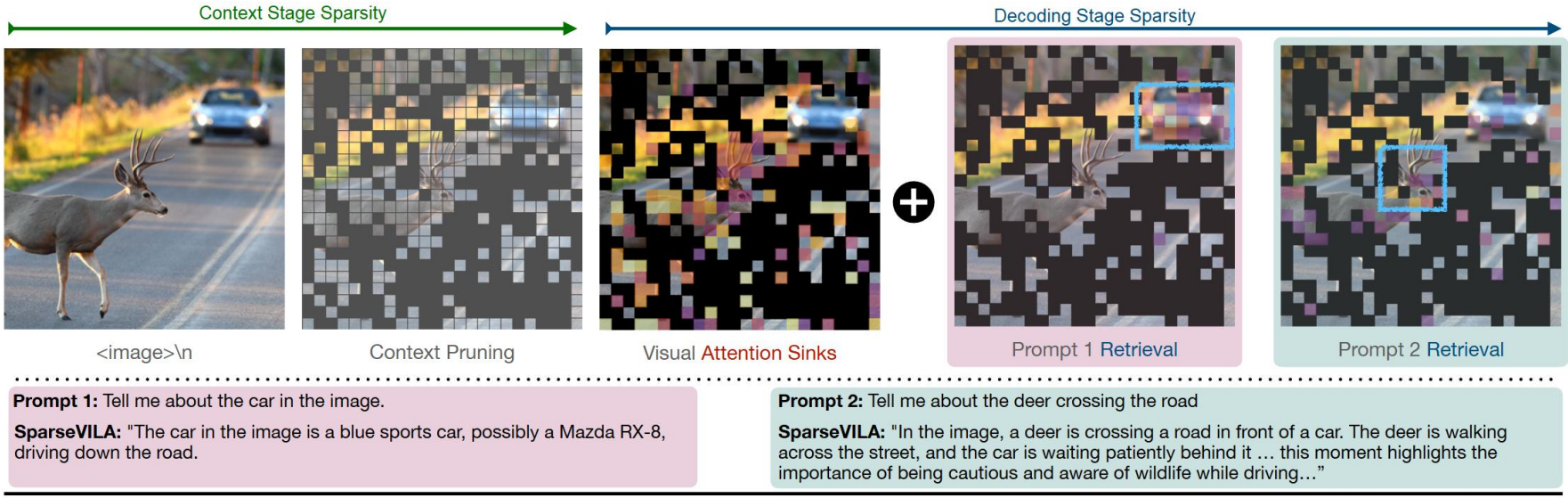
SparseVILA[3)]

- Method
  - Decoupled Prefill-Decode Visual Sparsity
    - Prefill sparsity (static)
      - 이미지 자체의 중복을 제거하며, 한 번 제거하면 영구적임 (보수적 pruning)
      - 메모리(KV cache) 절약에 기여
    - Decoding sparsity (dynamic)
      - 질문에 맞춰 필요한 정보만 필터링하며, 매 turn마다 다르게 적용
      - 추론 속도 향상에 기여



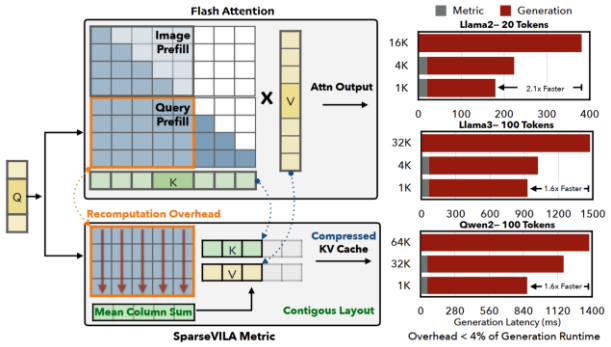
SparseVILA[3)]

- Method
  - Token categories analysis
    - Visual Attention Sinks
      - 질문과 상관없이 항상 중요한 토큰 (항상 유지)
    - Retrieval Tokens
      - 특정 질문에만 반응하는 토큰 (Query-aware 단계에서 선택)



SparseVILA[3)]

- Method
  - Efficient Implementation
    - FlashAttention2 사용
    - Dynamic recomputation & Compact KV Cache
      - 토큰을 pruning하여 sparse하면 비연속적 접근으로 인해 메모리도 속도가 느려짐
      - 중요한 토큰들을 골라서 dense하면서 compact한 새로운 KV cache를 만듦
    - Custom Triton Kernel
      - Attention를 계산하려면 모든 토큰 간의 관계를 계산해야하는데, 문맥이 길어지면 메모리가 터지거나 속도가 느려짐
      - Triton을 통해 코드 최적화
        - ✓OpenAI의 GPU 프로그래밍 언어(CUDA 대체)



SparseVILA[3)]

- Experiments
  - LLaVA-NeXT-7B

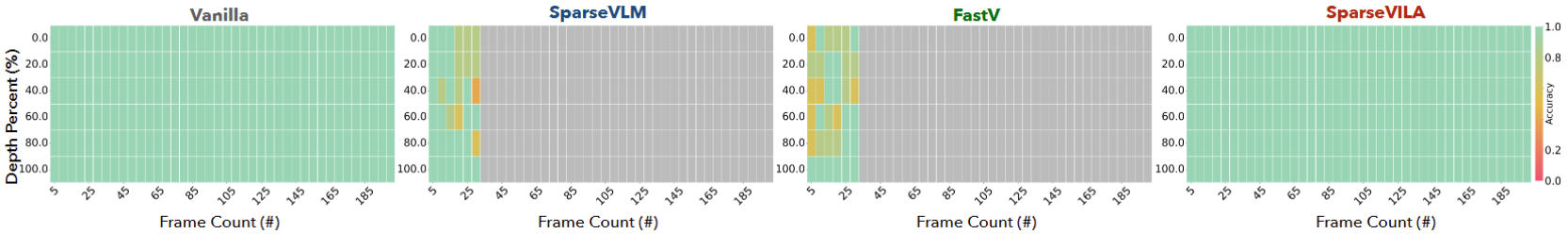
	Sparsity (%)		Speedup			CQA	DVQA	GQA	IVQA	MME	POPE	SQA	TVQA
	Prefill	Decode	Prefill	Decode	Overall								
LLaVA-NeXT-7B	–	–	–	–	–	50.7	74.4	64.2	37.7	1846.1	85.9	70.3	61.2
+ FastV	83	0	3.26×	1.24×	1.37×	37.3	54.4	60.6	31.8	1775.2	82.8	67.5	57.1
+ SparseVLM	83	0	3.26×	1.24×	1.37×	37.7	54.4	60.8	31.4	1780.7	84.0	67.5	58.1
+ PDrop	53	0	1.89×	1.16×	1.24×	41.5	44.8	62.9	26.5	1809.2	**86.1**	**69.1**	57.0
+ **SparseVILA**	67	50	2.59×	1.27×	1.38×	**41.8**	**60.8**	**63.4**	**33.3**	**1818.4**	85.9	68.0	**60.2**
+ PruMerge	83	0	4.34×	1.27×	1.43×	26.9	36.9	61.1	28.5	1753.2	83.6	66.7	54.6
+ HIRED	83	0	4.34×	1.27×	1.43×	29.7	41.6	62.4	28.2	1777.1	81.7	67.7	55.9
+ VisionZip	83	0	4.34×	1.27×	1.43×	34.1	51.2	61.2	30.6	1760.3	83.7	67.4	58.7
+ **SparseVILA**	67	90	2.59×	1.33×	1.44×	**39.9**	**54.4**	**63.4**	**30.8**	**1827.8**	**86.1**	**68.0**	**58.9**

  - Video

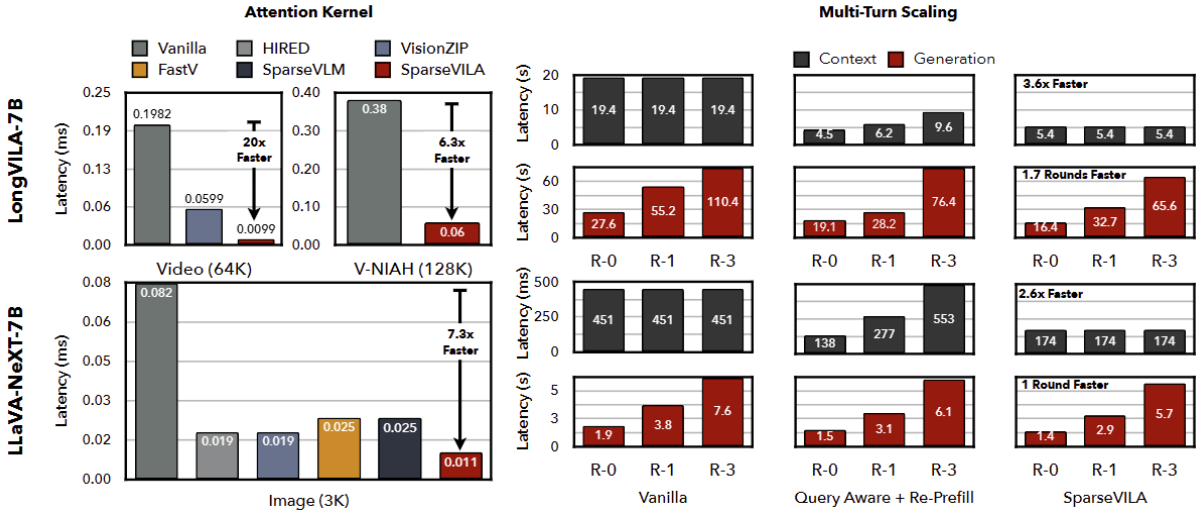
	Sparsity (%)		Speedup			MLVU	Video-MME (w/o subtitle)			
	Prefill	Decode	Prefill	Decode	Overall	m-avg	Short	Medium	Long	Overall
LongVILA-7B	–	–	–	–	–	72.4	68.9	58.1	52.0	59.7
+ HIRED	86	0	4.9×	1.5×	2.1×	70.2	68.0	54.6	49.6	57.4
+ VisionZip	86	0	4.9×	1.5×	2.1×	73.1	66.6	55.1	50.9	57.5
+ **SparseVILA**	75	94	3.6×	1.7×	2.2×	**74.4**	**70.6**	**58.6**	**51.7**	**60.3**

SparseVILA[3)]

- Experiments
  - Multi-turn



  - Efficiency Analysis



SparseVILA[3)]

- Conclusion
  - 기존 한계 극복
    - 기존 query-aware 방식이 cotext stage에서 정보를 영구적으로 삭제하여 발생하던 multi-turn 대화 시의 정보 손실 문제 해결
  - SparseVILA 제안
    - Visual sparsity 적용 시점을 prefill과 decoding 단계로 분리하는 새로운 decoupled framework를 제안
    - Prefill stage
      - 이미지 인코딩 단계에서는 시각적 중복을 제거하는 query-agnostic pruning을 적용하여 메모리 효율성을 확보
    - Decoding stage
      - 텍스트 생성 단계에서는 현재 질문에 필요한 토큰만 동적으로 retrieval하여 추론 속도를 가속

감사합니다