SparseVLM: Visual Token Sparsification for Efficient Vision-Language Model Inference

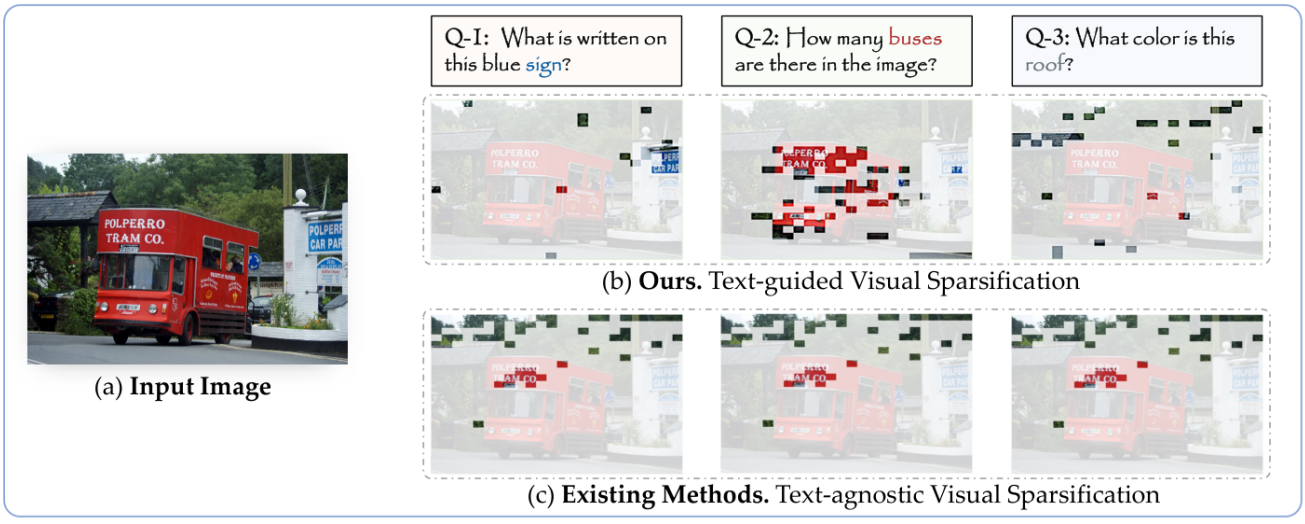
SparseVLM：面向高效视觉语言模型推理的视觉 token 稀疏化

导读摘要：这篇论文要解决的是 VLM 推理里一个非常现实的问题：视觉 token 太多，计算和显存代价太高，但真正对当前问题有用的图像信息往往只占一部分。作者提出 SparseVLM，用当前文本问题来指导视觉 token 的保留与剪枝，而且整个过程不需要额外训练。它的关键不只是“删 token”，而是先筛出真正和图像相关的文本 token 作为评分者，再按层自适应决定删多少，并把被删 token 中仍有价值的信息重新压成更小表示。实验上，这种做法在 LLaVA、Mini-Gemini、Qwen2-VL 和 VideoLLaVA 上都取得了更好的精度—效率平衡，尤其在高压缩场景下优势明显。

研究背景：为什么视觉 token 需要“按问题裁剪”

视觉语言模型的一个核心瓶颈，是视觉序列通常远长于文本序列。高分辨率图像、长视频、多帧输入都会把 token 数迅速推高，最终让注意力计算、KV cache 和延迟都成为部署障碍。论文的出发点并不复杂：图像包含大量空间信息，但对具体问题真正有用的区域通常很少，因此不应把所有视觉 token 一视同仁地送进解码器。

更关键的是，作者认为很多已有压缩方法虽然也在减少视觉 token，却往往**不看用户到底问了什么**。这在多模态问答里其实很不合理：问路牌文字、问公交数量、问屋顶颜色，显然应该保留完全不同的图像区域。论文用一个很直观的例子说明了这一点：



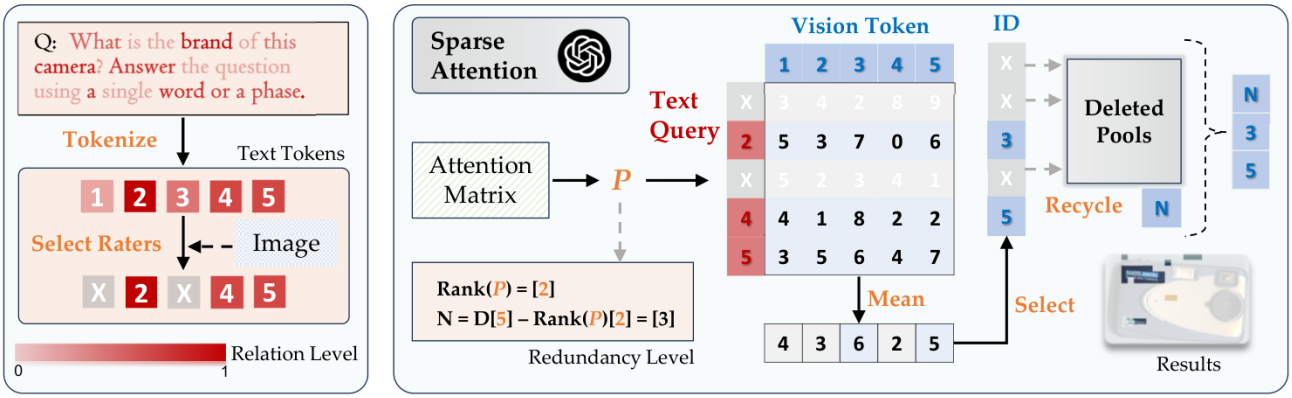
这张图几乎就是整篇论文的问题定义：同一张图，不同问题对应的“重要视觉区域”应该不同。SparseVLM 的目标因此不是做静态压缩，而是做**文本感知的动态稀疏化**。这也解释了为什么作者后面的实验重点，不只是看 FLOPs 降多少，更看在同等延迟或同等 token 预算下，回答质量能保住多少。

方法主线：文本先决定“谁来评分”，再决定“删多少、怎么补”

SparseVLM 的方法设计有三个连续动作：

- 先从问题里挑出真正和图像相关的文本 token，作为视觉 token 的“评分者”；
- 在解码过程中，利用已有自注意力矩阵估计视觉 token 对这些评分者的贡献，并按层自适应稀疏化；
- 对被剪掉但仍有价值的 token 做回收和重构，减少信息损失。

整体流程可以从这张总览图快速把握：



(a) Before LLM (b) During LLM

这套方法最巧的地方，是它尽量复用模型已经算出来的东西。作者不训练新的裁剪网络，而是直接使用解码器中的自注意力关系，从“文本查询—视觉键”那一部分估计哪些视觉 token 更重要。这样做的好处很明确：训练免费、可插拔、额外参数几乎没有。

但作者并没有停在“用文本指导视觉”这么粗的层面。论文特别强调，**不是所有文本 token 都适合参与评分**。问题中的介词、代词、功能词，甚至某些与图像弱相关的词，都会给视觉评分带来噪声。因此他们先在进入解码器前，根据图文相关性筛出一批更像关键词的文本 token，即 text raters。之后才让这些 raters 去评价视觉 token 的重要性。

在确定“谁重要”之后，SparseVLM 还要解决“每层到底删多少”。这里论文使用了一个基于文本—视觉关系矩阵秩的策略：如果矩阵冗余更大，就说明当前视觉表示里可压缩成分更多，可以删得更激进；如果冗余小，则少删甚至不删。相比给所有层固定比例，这种按层自适应更符合不同层语义密度的变化。

最后还有一个很实用的设计：**token recycling**。不少方法把不重要的视觉 token 直接丢掉，SparseVLM 则会从已删 token 中挑出仍有一定价值的一部分，聚合后重构成更紧凑的新 token，再送回模型。这个设计在高压缩率下尤其关键，因为它能解释为什么 SparseVLM 在只保留很少 token 时，性能仍比直接裁剪法稳得多。

评测设定：覆盖图像、视频与跨模型泛化

实验部分相当全面，而且明显围绕“效率—性能折中”展开。

图像理解任务上，作者在 8 个常见基准上评测，包括 GQA、MMBench、MME、POPE、SQA、SEED、TextVQA 和 MMVet。模型方面覆盖了三类主流 VLM：

- **LLaVA**
- **Mini-Gemini**
- **Qwen2-VL**

这意味着论文并不只想证明“某个模型上能用”，而是强调 SparseVLM 作为推理阶段模块的通用性。

视频理解任务上，作者把方法直接扩展到 VideoLLaVA，并在 TGIF-QA、MSVD-QA、MSRVTT-QA、ActivityNet-QA 上测试。这里的重点不是重新设计视频结构，而是验证：如果输入带有时间维度、视觉 token 规模更大，文本引导稀疏化是否仍然有效。

整体评测维度主要有三类：

- **任务准确率/得分**
- **FLOPs**
- **实际 CUDA 延迟与显存收益**

这点很重要，因为许多压缩论文只报告理论压缩率，但 SparseVLM 明确把真实推理代价也纳入比较。尤其在与 ToMe、FastV、PDrop 的对比中，作者希望证明的是：不是“压得更多”就好，而是要在相近代价下保留更高性能。

核心实验结果：高压缩下依然保持更强性能

论文最核心的主表来自 LLaVA。在这里，原始视觉 token 数为 576，作者分别测试保留 192、128、64 个 token 的情形，也就是从中等压缩到极高压缩。

first line of each method is the raw accuracy of benchmarks, and the second line is the proportion relative to the upper limit.

Method	GQA	MMB	MME	POPE	SQA	SEED	VQA^{Text}	MMVet	Acc. (%)	FLOPs (T)	Latency (ms)
Upper Bound, 576 Tokens **(100%)**											
Vanilla	61.9 100%	64.6 100%	1864 100%	85.9 100%	69.5 100%	60.3 100%	58.3 100%	30.9 100%	100	4.62	57.82
Retain 192 Tokens (↓ **66.7%**)											
ToMe (ICLR23)	54.3 87.7%	60.5 93.5%	1563 83.9%	72.4 84.3%	65.2 93.8%	53.1 88.1%	52.1 89.5%	27.9 90.3%	88.9 (↓ 11.1)	2.05	34.06
FastV (ECCV24)	52.6 85.0%	61.0 94.4%	1605 86.1%	64.8 75.4%	69.1 99.4%	52.1 86.4%	52.5 90.1%	26.7 86.4%	87.9 (↓ 12.1)	2.11	34.87
PDrop (CVPR25)	57.1 92.2%	63.2 97.8%	1766 94.7%	82.3 95.8%	70.2 101.0%	54.7 90.7%	56.1 96.2%	30.5 98.7%	95.9 (↓ 4.1)	2.03	36.74
SparseVLM	59.5 96.1%	64.1 99.2%	1787 95.9%	85.3 99.3%	68.7 98.8%	58.7 97.3%	57.8 99.1%	33.1 107.1%	99.1 (↓ **0.9**)	2.14	36.50
Retain 128 Tokens (↓ **77.8%**)											
ToMe (ICLR23)	52.4 84.7%	53.3 82.4%	1343 72.1%	62.8 73.1%	59.6 85.8%	50.9 84.4%	49.1 84.4%	27.2 88.0%	81.9 (↓ 18.1)	1.62	30.00
FastV (ECCV24)	49.6 80.1%	56.1 86.8%	1490 79.9%	53.4 62.2%	68.6 98.7%	48.1 79.8%	50.5 86.6%	26.3 85.1%	82.4 (↓ 17.6)	1.70	30.70
PDrop (CVPR25)	56.0 90.5%	61.1 95.4%	1664 89.3%	82.3 95.8%	69.9 100.6%	53.3 88.4%	55.1 94.5%	30.8 99.7%	94.3 (↓ 5.7)	1.62	37.77
SparseVLM	58.4 94.3%	64.5 99.8%	1746 93.7%	85.0 99.0%	68.6 98.7%	58.2 96.5%	56.7 97.3%	29.0 93.9%	96.7 (↓ **3.3**)	1.72	33.28
Retain 64 Tokens (↓ **88.9%**)											
ToMe (ICLR23)	48.6 78.5%	43.7 67.5%	1138 61.1%	52.5 61.1%	50.0 71.9%	44.0 73.0%	45.3 77.8%	24.1 78.0%	71.1 (↓ 28.9)	1.19	26.52
FastV (ECCV24)	46.1 74.5%	47.2 73.1%	1255 67.3%	38.2 44.5%	68.7 98.8%	43.7 72.5%	47.8 82.0%	19.6 63.4%	72.0 (↓ 28.0)	1.29	27.30
PDrop (CVPR25)	41.9 67.7%	33.3 51.6%	1092 58.6%	55.9 65.1%	69.2 99.6%	40.0 66.3%	45.9 78.7%	30.7 99.4%	73.4 (↓ 26.6)	1.18	43.41
SparseVLM	53.8 86.9%	60.1 93.0%	1589 85.2%	77.5 90.2%	69.8 100.4%	52.2 86.6%	53.4 91.6%	24.9 80.6%	89.3 (↓ **10.7**)	1.30	29.89

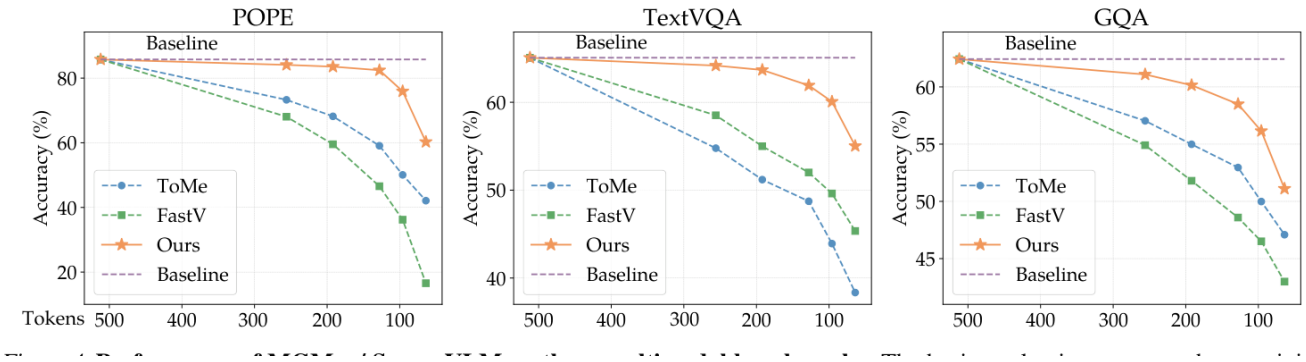
这张表非常值得细看。它传递出几个清晰结论：

第一，**在中等压缩下，SparseVLM 几乎不伤性能**。当 576 个视觉 token 压到 192 个时，平均准确率只下降 0.9%，但 FLOPs 从 4.62T 降到 2.14T，延迟也明显下降。也就是说，模型大约用不到一半的计算，保住了约 99% 的整体性能。

第二，**压缩越激进，SparseVLM 相对优势越大**。在只保留 64 个 token 时，SparseVLM 的平均表现仍显著优于 ToMe、FastV 和 PDrop。尤其与 FastV 相比，作者报告在类似延迟下可带来 11.2% 到 17.3% 的优势。这个现象很关键，因为真正有部署价值的方法，往往要在极低 token 预算下仍不“崩”。

第三，**SparseVLM 的收益不只体现在平均值**。从各个任务列看，它在 GQA、MMBench、POPE、TextVQA 等任务上都更稳，说明文本感知稀疏化并不是对某一类数据集特化，而是普遍有帮助。尤其对需要定位细粒度证据的任务，问题相关区域被更准确保留，往往就是性能差异的来源。

作者还在 Mini-Gemini 上用曲线图展示了不同 token 数下的精度变化趋势，这比单点对比更能看出方法的稳定性：



这张图的含义非常直接：随着保留 token 数持续下降，SparseVLM 的准确率曲线始终整体高于 ToMe 和 FastV，并更贴近基线。更重要的是，在低 token 区域，曲线差距被明显拉开。这说明 SparseVLM 的优势不是偶然波动，而是在压缩更强时更稳定地体现出来。作者将其归因于两点：一是 text-aware 的筛选能更准确定位任务相关 patch；二是 recycling 减少了激进裁剪带来的信息损失。

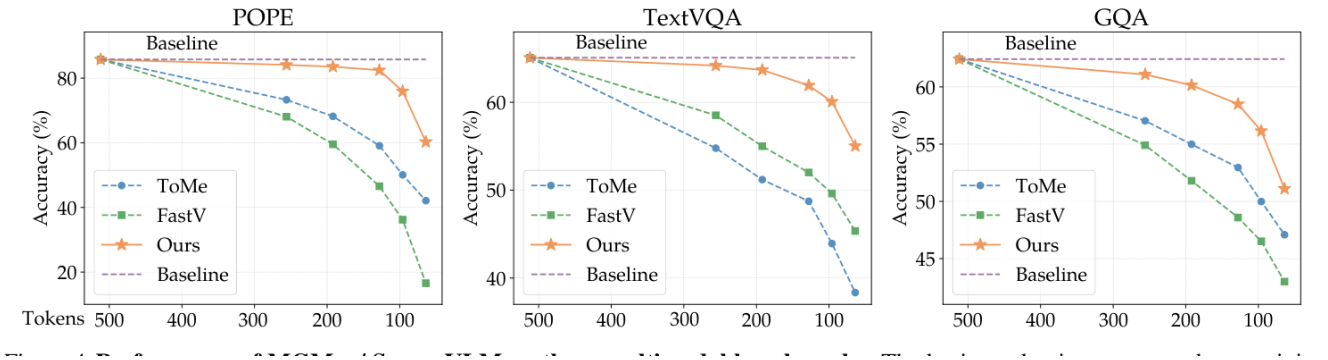
跨模型泛化方面，Qwen2-VL 的结果也很有说服力：

first line of each method is the raw accuracy of benchmarks, and the second line is the proportion relative to the upper limit.

Method	GQA	MMB	MME	POPE	SQA	SEED	VQA^{Text}	MMVet	Acc. (%)	FLOPs (T)	Latency (ms)
Upper Bound, 576 Tokens **(100%)**											
Vanilla	61.9 100%	64.6 100%	1864 100%	85.9 100%	69.5 100%	60.3 100%	58.3 100%	30.9 100%	100	4.62	57.82
Retain 192 Tokens (↓ **66.7%**)											
ToMe (ICLR23)	54.3 87.7%	60.5 93.5%	1563 83.9%	72.4 84.3%	65.2 93.8%	53.1 88.1%	52.1 89.5%	27.9 90.3%	88.9 (↓ 11.1)	2.05	34.06
FastV (ECCV24)	52.6 85.0%	61.0 94.4%	1605 86.1%	64.8 75.4%	69.1 99.4%	52.1 86.4%	52.5 90.1%	26.7 86.4%	87.9 (↓ 12.1)	2.11	34.87
PDrop (CVPR25)	57.1 92.2%	63.2 97.8%	1766 94.7%	82.3 95.8%	70.2 101.0%	54.7 90.7%	56.1 96.2%	30.5 98.7%	95.9 (↓ 4.1)	2.03	36.74
SparseVLM	59.5 96.1%	64.1 99.2%	1787 95.9%	85.3 99.3%	68.7 98.8%	58.7 97.3%	57.8 99.1%	33.1 107.1%	99.1 (↓ **0.9**)	2.14	36.50
Retain 128 Tokens (↓ **77.8%**)											
ToMe (ICLR23)	52.4 84.7%	53.3 82.4%	1343 72.1%	62.8 73.1%	59.6 85.8%	50.9 84.4%	49.1 84.4%	27.2 88.0%	81.9 (↓ 18.1)	1.62	30.00
FastV (ECCV24)	49.6 80.1%	56.1 86.8%	1490 79.9%	53.4 62.2%	68.6 98.7%	48.1 79.8%	50.5 86.6%	26.3 85.1%	82.4 (↓ 17.6)	1.70	30.70
PDrop (CVPR25)	56.0 90.5%	61.1 95.4%	1664 89.3%	82.3 95.8%	69.9 100.6%	53.3 88.4%	55.1 94.5%	30.8 99.7%	94.3 (↓ 5.7)	1.62	37.77
SparseVLM	58.4 94.3%	64.5 99.8%	1746 93.7%	85.0 99.0%	68.6 98.7%	58.2 96.5%	56.7 97.3%	29.0 93.9%	96.7 (↓ **3.3**)	1.72	33.28
Retain 64 Tokens (↓ **88.9%**)											
ToMe (ICLR23)	48.6 78.5%	43.7 67.5%	1138 61.1%	52.5 61.1%	50.0 71.9%	44.0 73.0%	45.3 77.8%	24.1 78.0%	71.1 (↓ 28.9)	1.19	26.52
FastV (ECCV24)	46.1 74.5%	47.2 73.1%	1255 67.3%	38.2 44.5%	68.7 98.8%	43.7 72.5%	47.8 82.0%	19.6 63.4%	72.0 (↓ 28.0)	1.29	27.30
PDrop (CVPR25)	41.9 67.7%	33.3 51.6%	1092 58.6%	55.9 65.1%	69.2 99.6%	40.0 66.3%	45.9 78.7%	30.7 99.4%	73.4 (↓ 26.6)	1.18	43.41
SparseVLM	53.8 86.9%	60.1 93.0%	1589 85.2%	77.5 90.2%	69.8 100.4%	52.2 86.6%	53.4 91.6%	24.9 80.6%	89.3 (↓ **10.7**)	1.30	29.89

Qwen2-VL 本身带有动态分辨率特性，因此这组实验说明 SparseVLM 不依赖固定 token 结构。表中可以看到，从动态原始设置到 600、500、400 token，平均精度是平缓下降的，没有出现压缩后断崖式恶化。作者特别提到，去掉 54.5% 的视觉 token 后，仍能保留约 98% 的原始性能。这说明方法并不局限于 LLaVA 系列，而能兼容更实用的高分辨率 VLM。

视频任务的结果进一步增强了论文说服力，因为视频通常比单图更难压缩：既有空间信息又有时间信息，token 数也更大。

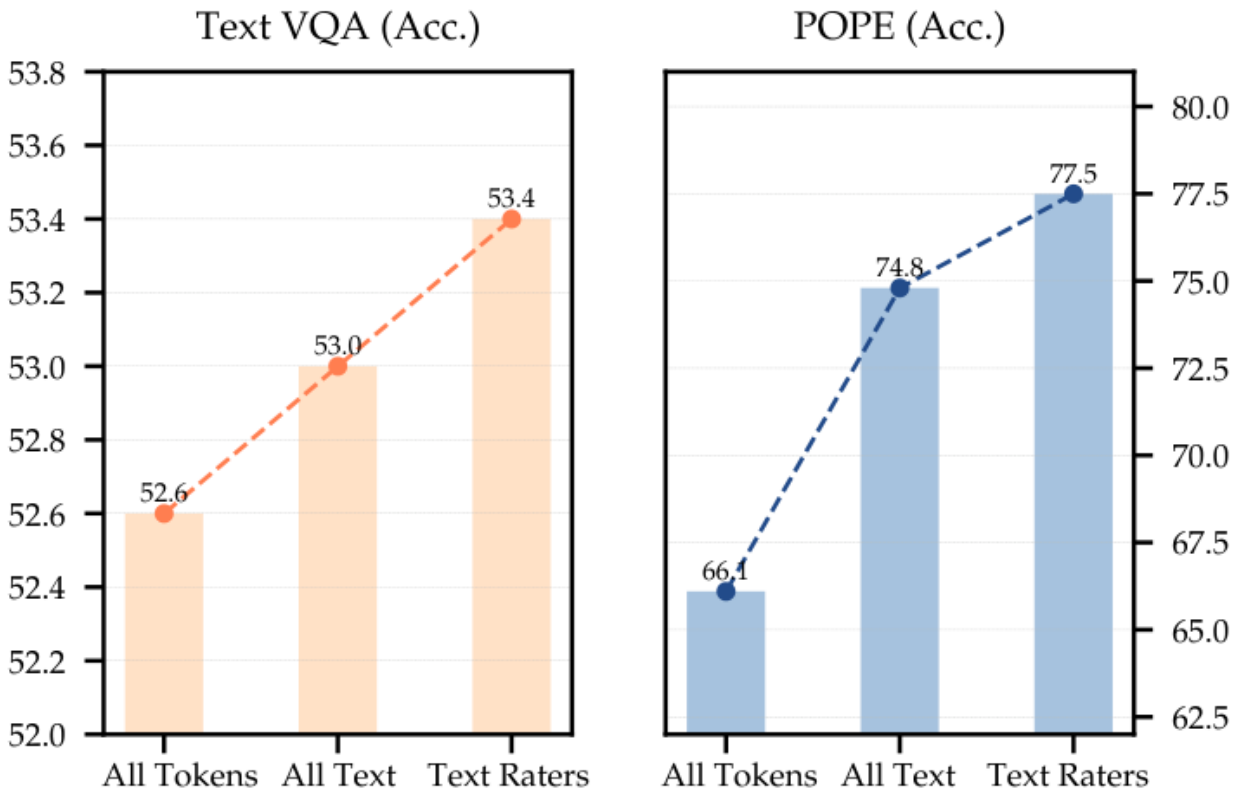


这里原始 VideoLLaVA 使用 2048 个视频 token，而实验把它们压到 194 个，压缩比例高达 90.5%。在这种极端设置下，SparseVLM 仍明显优于 FastV，平均准确率接近原始模型，而 GPT 评分损失也很小。论文总结为：在相同 token 预算下，SparseVLM 比 FastV 平均高 14.7%。这说明“按问题保留什么帧、什么区域”的策略，对视频同样有效，而不仅仅是图像问答中的特例。

消融与结果解读：为什么它比“粗暴删 token”更稳

如果只看主结果，可以知道 SparseVLM 有效；而消融实验进一步说明，它到底为什么有效。

首先是最核心的 text raters 设计。作者比较了三种评分依据：使用所有 token、只用文本 token、以及进一步筛选后的 text raters。



这个消融说明两层结论。一是“文本引导”本身就比无差别使用所有 token 更好；二是“并非所有文本都该参与引导”，进一步筛出与图像真正相关的 raters 才最好。例如在 POPE 上，text raters 相比只用全部文本 token 还有明显提升，说明某些任务对提示词噪声非常敏感。这也从实验上支撑了论文的方法细节：真正重要的不是“让文本参与”，而是“让正确的文本参与”。

其次是被剪 token 的重构机制。论文在 LLaVA 7B 上比较是否加入 token reconstruction：

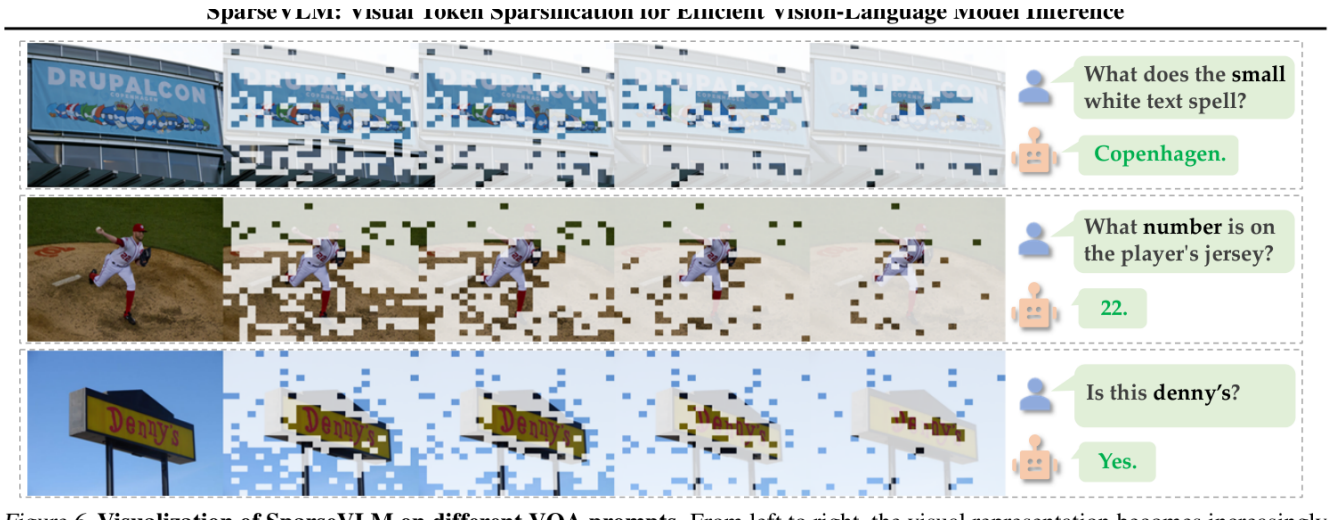


Figure 6. Visualization of SparseVLM on different VQA prompts. From left to right, the visual representation becomes increasingly

这张表的趋势很稳定：加入重构后，GQA 和 POPE 上几乎都更好，而且压缩越狠，收益越明显。也就是说，recycling 并不是可有可无的小修饰，而是在高压缩率下维持性能的关键部件。原因也很好理解：当大量 token 被删时，直接丢弃会让细节损失迅速累积，而把其中有价值的一部分重新压回少量 token，可以显著缓解这种损失。

最后，论文还给出了一组很有解释性的可视化结果：

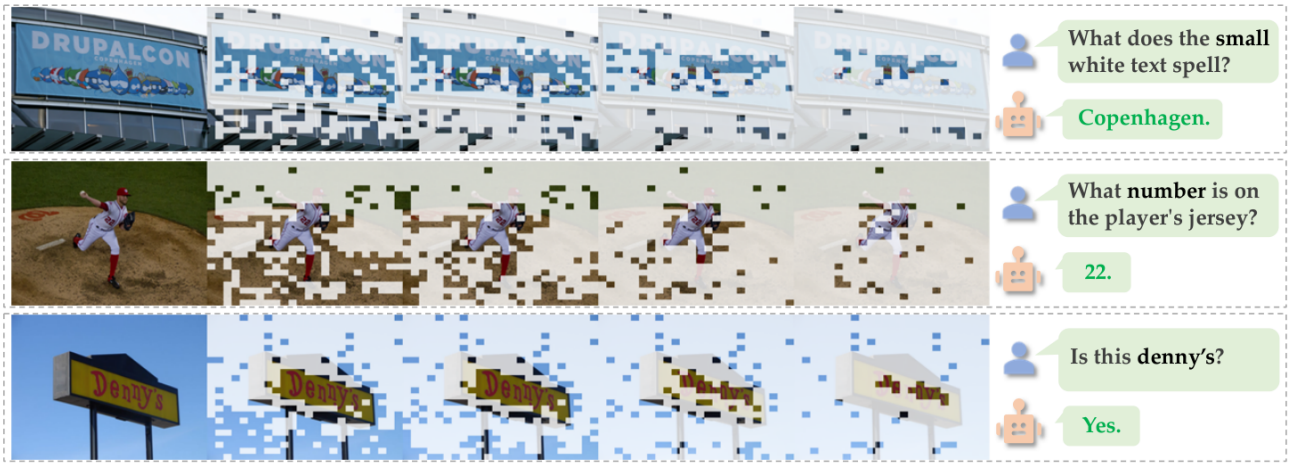


Figure 6. Visualization of SparseVLM on different VQA prompts. From left to right, the visual representation becomes increasingly

从这些例子里可以直观看到，随着层数推进、视觉表示越来越稀疏，保留下来的区域越来越集中在与问题直接相关的位置：招牌小字、球衣号码、店名等。这种现象说明 SparseVLM 不是均匀地“缩图”，也不是盲目地保留显著前景，而是在逐步收缩到回答当前问题所需的证据区域。这正好呼应了前面的定量结果：之所以能在极少 token 下保住精度，不是因为模型变“更会猜”，而是因为关键视觉证据被更可靠地留下来了。

总结：一篇把“问题相关性”真正引入视觉压缩的工作

SparseVLM 的贡献，表面上看是训练免费的视觉 token 稀疏化；更深一层看，它把“视觉压缩该由当前问题来决定”这件事做成了一个简单、可插拔、跨模型可复用的推理机制。

从实验角度，这篇论文最强的点有三个：

- **压缩强度高**：图像和视频都能大幅减少视觉 token；
- **性能保持稳**：尤其在极高压缩率下，明显优于 ToMe、FastV 等方法；
- **泛化范围广**：覆盖 LLaVA、Mini-Gemini、Qwen2-VL、VideoLLaVA，不局限于单一骨干。

如果只记一个结论，可以记成一句话：**SparseVLM 不是简单少看图像，而是按问题有选择地看图像，因此在同样更快的前提下，通常比不看问题的压缩方法保留了更多正确答案所需的信息。**