VLM中自适应压缩率的视觉token剪枝

1. 研究背景

文章关注的问题

VLMs 需融合视觉与文本信息完成问答、图像描述等任务，其输入的token序列N较大，由于VLMs 本身的自回归生成（输出也作为输入） 与注意力二次复杂度（O (N^2)） 特性，形成 “叠加放大效应”，使得计算量和内存占用量过大，限制了VLMs的实际应用（高分辨率图像等）。

现有方法的局限

非token缩减技术类

- MoELlava 集成混合专家（Mixture of Experts）框架以加速模型运行；
- VL-Mamba探索替代架构来提升效率；
- LLaVA-Phi、mobileVLM采用更小的语言模型，这类模型能在性能损失极小的前提下实现高效运行。
- 此外，剪枝 、量化、知识蒸馏等压缩技术也被广泛用于减少模型参数数量。

局限性：需要修改模型架构或参数，不便于后续开发

相比之下，token缩减技术无需改变模型架构，仅通过精简token序列即可解决 VLMs 的二次复杂度问题。

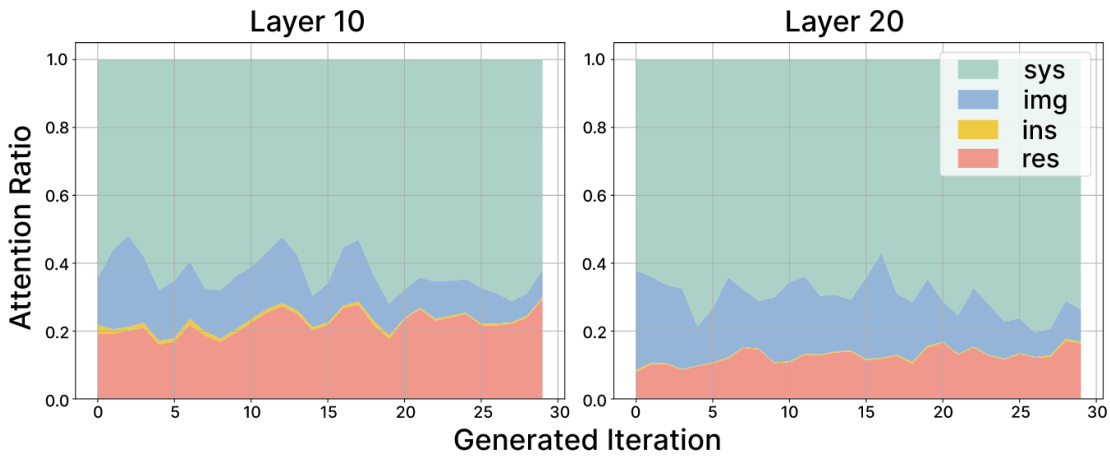
token缩减技术

- 视觉编码器层面：LLaVA-PruMerge、MADTP 等提出自适应方法减少视觉token，在大幅降低token数量的同时，保持与原始模型相当的性能；
- 跨模态投影层层面：Tokenpacker 优化文本与视觉信息的连接桥梁，以最小化视觉token数量；
- 推理阶段层面：FastV一文发现，视觉token在解码器第二层之后获得的关注度会降低，因此选择在推理过程中直接移除冗余视觉token，在不影响性能的前提下降低计算需求；VTW 发现视觉token在 VLMs 深层中的作用并不显著，因此策略性地从特定层中移除所有视觉token，仅允许文本token参与后续处理。

局限性：

- FastV等方法按预定义的压缩率删除冗余视觉token，而这里的**压缩率需要手动指定**。而手动选择合适的压缩率并非易事，通常需要专家级的领域知识。

- 预定义的**压缩率是一个固定值**，而实验证明（如下图），随着模型生成过程的推进，视觉令牌的重要性会逐渐降低。根据生成过程中变化的注意力分布动态调整压缩率会是一种更好的选择，这也是本文最大的创新点。

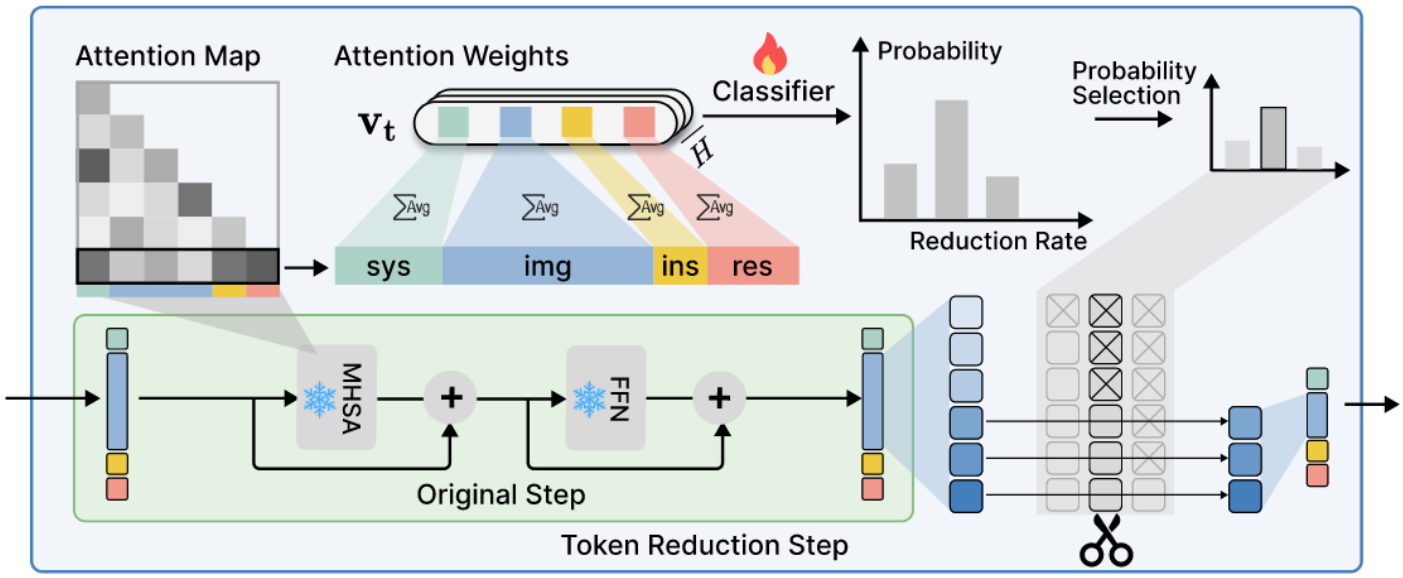


实验结果表明：随着模型生成的令牌数量增多，其注意力会向响应令牌倾斜，而对视觉令牌的关注度逐渐降低，这意味着视觉令牌的冗余性不断增加。

根据信息流理论（少量 “锚定令牌”（anchor tokens）聚合了所有输入令牌的信息，而模型在深层网络中更倾向于关注这些锚定令牌），生成过程中图像令牌的大量信息会聚合到响应令牌中。这种信息聚合导致视觉令牌之间产生额外冗余 —— 大量视觉令牌仅提供极少有效信息，造成计算资源的浪费。

2. 本文方法

整体框架

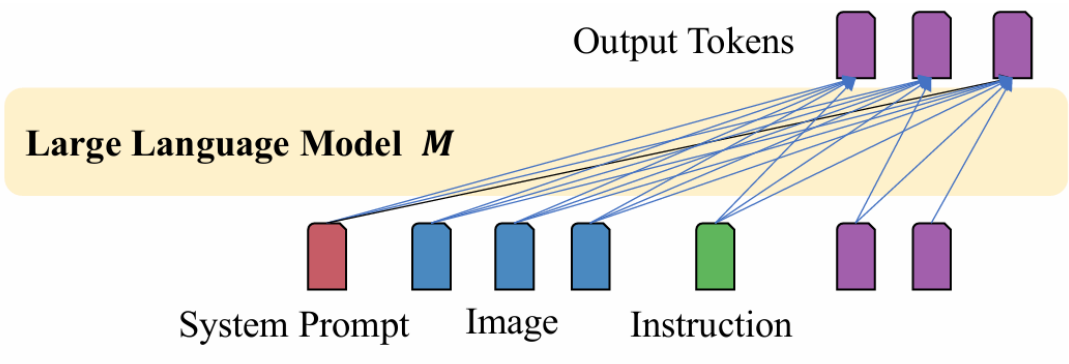


总体采用端到端的训练。

动态剪枝模块

将输入的token分为四类：

- 系统提示(sys)
- 图像(img)
- 用户指令(ins)
- 响应(res)



假定视觉令牌的冗余度与上述四种token的注意力分布密切相关，在模型生成的每一轮迭代，收集所有注意力头对 sys、img、ins、res 四类 token 的注意力权重，将这些权重统计为 “四类 token 的关注占比向量”（如 img 占比 30%、res 占比 60%、sys 占比 5%、ins 占比 5%），同时将img类中的每个token按注意力权重排序。

将占比向量作为训练输入，输出压缩率的概率分布，选择概率值最大的reduction rate作为采纳值R。

然后以R的比率去除img类中注意力权重较低的token，得到最终的img token。

即：对于每一组 token类占比四元组（也即每一次迭代），都有一个压缩率与之对应，实现了动态视觉token剪枝的效果。

可微压缩率

目的：为了能使梯度回传，进行端到端训练，采样和掩码操作会丧失可微性。

1）将压缩率 R 离散化为 $\mathcal{R}:\{0,\frac{1}{K},\frac{2}{K},\ldots,\frac{K-1}{K}\}$

2) 为每一个 R_i 配备一个掩码向量 $M_i=[1,1,1,\ldots,0]$

$M_0=[1,\ldots,1,1,1](r=0)$

$M_1=[1,\ldots,1,0,0](r=\frac{1}{K})$

$M_2=[1,\ldots,0,0,0](r=\frac{2}{K})$

$\ldots$

$M_K=[0,\ldots,0,0,0](r=\frac{K-1}{K})$

3) 给一个描述4类token分布的特征向量，经过classifier，输出一个概率分布 π_R ，

再将 π_R 送入 $Gumbel-Softmax(\pi_R)$,得到如下软化的概率分布

$$P(r=0)=p_1$$

$$P(r=\frac{1}{K})=p_2$$

$$P(r=\frac{2}{K})=p_3$$

$$\cdots$$

$$P(r=\frac{K-1}{K})=p_K$$

4)加权求和得到最终的掩码向量：

$$M=\sum M_i \cdot p_{i+1}$$

M 与image token特征向量相乘得到最终剪枝后的结果

3. 实验结果分析

对比实验

视觉问答任务与短响应任务

从上到下分为三个对比区域：

- 早期 “独立设计型” VLM
- 基于 LLaVA 生态的 “开源衍生型” VLM
- “令牌优化型” VLM

TABLE I

COMPARISON AMONG DIFFERENT VLMS ON 4 VISUAL QUESTION ANSWERING BENCHMARKS AND 3 COMMON BENCHMARKS. BENCHMARK NAMES ARE ABBREVIATED DUE TO SPACE LIMITS. THE HIGHEST-PERFORMING RESULTS ARE HIGHLIGHTED IN **BOLDFACE**.

Model	LLM	Res.	Visual Question Generation				Short Generation		
			GQA↑ *EM*	VisWiz↑ *EM*	SQA^{I}↑ *EM*	VQA^{T}↑ *EM*	POPE↑ *Accuracy*	MMB↑ *Accuracy*	MME↑ *Accuracy*
InstructBLIP	Vicuna-7B	224	49.2	34.5	60.5	50.1	79.8	36.0	–
InstructBLIP	Vicuna-13B	224	49.5	33.4	63.1	50.7	78.9	–	1212.8
MiniGPT-4	Vicuna-13B	224	41.0	19.6	61.0	42.5	85.3	–	1293.8
Qwen-VL	Qwen-7B	448	59.3	35.2	67.1	63.8	–	38.2	–
Qwen-VL-Chat	Qwen-7B	448	57.5	38.9	68.2	61.5	–	60.6	1487.5
LLaMA-VID	Vicuna-7B	336	64.3	54.1	68.3	–	86.0	63.4	1521.4
VoCo-LLaMA	Vicuna-7B	33	57.0	53.0	65.4	52.7	81.4	58.8	1323.3
TokenPacker	Vicuna-7B	336	61.9	52.0	–	–	87	65.1	–
M^3	Vicuna-7B	336	61.3	53.1	67.2	–	86.6	63.6	–
PruMerge	Vicuna-7B	336	61.8	53.5	68.5	56.0	76.3	60.9	1350.3
LLaVAv1.5	Vicuna-7B	336	62.0	50.0	66.8	58.2	85.9	64.3	1510.7
FastV	Vicuna-7B	336	60.3	**54.4**	69.0	45.4	82.5	63.9	1510.2
VTW	Vicuna-7B	336	55.1	50.9	69.1	16.1	85.9	64.0	1501.4
SparseVLM	Vicuna-7B	336	57.6	–	69.1	56.1	83.6	62.5	1721
VisionZip	Vicuna-7B	336	60.1	–	68.9	57./	84.93	63.4	**1834**
DyRate(ours)	Vicuna-7B	336	**61.9**	54.2	**69.2**	**45.7**	**86.8**	**64.1**	1516.6

视觉问答任务benchmark：

- GQA(复杂结构化推理)：与传统模型匹敌
- VisWiz(视觉障碍辅助、细粒度细节):比FastV弱，但已经很好
- SQA (多轮序列式推理)：与传统模型匹敌
- VQA v2 (通用视觉问答)：偏弱（文章好像标错了）
- POPE(衡量模型物体幻觉)：优于传统模型，很强
- MMB（对象级感知任务）：与传统模型匹敌
- MME(多模态综合能力)：与传统模型匹敌

图像描述任务

TABLE II

CIDER SCORES OF DIFFERENT METHODS ON NOCAPS, FLICKR30K, AND COCO2017 DATASETS.

Models	Methods	Nocaps↑ *CIDEr*	Flickr30k↑ *CIDEr*	COCO2017↑ *CIDEr*
LLaVA-1.5 -7B	Original	74.89	105.57	110.43
	$FastV_{(K=3,R=0.5)}$	74.75	105.00	110.80
	$FastV_{(K=2,R=0.5)}$	74.86	104.00	110.40
	$VTW_{(K=16,R=1)}$	44.54	58.00	67.20
	DyRate(*ours*)	75.00	108.41	110.54
LLaVA-1.5 -13B	Original	109.31	79.56	115.57
	$FastV_{(K=3,R=0.5)}$	102.70	73.40	105.36
	$FastV_{(K=2,R=0.5)}$	103.10	73.40	108.85
	$VTW_{(K=16,R=1)}$	95.21	65.87	101.67
	DyRate(ours)	113.23	79.38	115.72

DyRate方法在图像描述任务上优于不剪枝和其他token剪枝方法

复杂场景描述任务

TABLE III
COMPARING TOKEN REDUCTION METHODS FOR COMPLEX SCENE DESCRIPTION.

Methods	TFLOPs(%)↓	Latency (ms)↓	Nocaps↑	Auto R
LLaVA-1.5-7B				
Original	100.0	70.80	74.89	✗
$\text{FastV}_{(K=3,R=0.5)}$	57.90	42.36	74.75	✗
$\text{FastV}_{(K=2,R=0.5)}$	55.40	41.37	74.86	✗
$\text{VTW}_{(K=16,R=1)}$	55.20	46.24	44.54	✗
DyRate(ours)	33.33	40.13	75.00	✓
LLaVA-1.5-13B				
Original	100.0	128.36	109.31	✗
$\text{FastV}_{(K=3,R=0.5)}$	57.90	73.35	102.70	✗
$\text{FastV}_{(K=2,R=0.5)}$	55.40	72.25	103.10	✗
$\text{VTW}_{(K=16,R=1)}$	55.20	80.78	95.21	✗
DyRate(ours)	33.33	65.16	113.23	✓

TFLOPs(算法复杂度)和延迟:均最低

理解能力：最高

综上，对比实验表明，DyRate方法剪枝token后的模型不仅降低了计算开销，理解能力在大部分指标下依旧保持甚至略有提升。总体效果不逊色于FastV方法。

消融实验

压缩率策略：

消融实验探究了不同剪枝策略对 VLMs 性能的影响，为保证对比有效性，剪枝层数设置为 K=3（与 FastV 一致）。

固定压缩率策略（FixedPrune, FP）：

$$C_{retrain} = 1 - R$$

基于层深度动态调整压缩率的策略（DepthBasedPrune, DP）：

- 当 $L_{index} \leq 4$（前 4 层）：$H(L_{index} - 4)$ =0，因此 $C_{retrain}$ =1 → 不剪枝，保留所有视觉令牌。
- 当 $L_{index} > 4$（第 5 层及更深层），$H(L_{index} - 4)$，因此 $C_{retrain} = 1 - P_{prune4th} - R'$ → 从第 5 层开始，按照 “第 4 层剪枝比例 + 修正比例” 来剪枝，保留的令牌比例减少。

(b) **Pruning Strategy**

Strategy	Nocaps↑ *CIDEr*	FLOPs↓ (%)
LLaVA	105.57	100.00
FastV	105.00	57.90
Ours(FP)	107.00	28.80
Ours(DP)	105.00	58.10

（怀疑FP和DP是不是弄反了）

说明较大数目的剪枝不影响性能。

解码参数

Greedy（只选择模型预测概率最高的那个词）

Beem-K（Top-K 个概率最高的候选词序列）

top p（先将所有候选词按概率从高到低排序，再累积概率直到达到阈值p，仅从这个 “概率核” 中随机采样下一个词。）

(a) **Generation Type**

Strategy	Nocaps↑ *CIDEr*
Greedy	106.74
Beam-2	109.06
Beam-5	108.28
top_p	**109.57**

top_p策略最优

4. 一些想法

没有一些成体系的想法，就把读论文时的一些杂念写进去了。

感觉这篇论文就是FastV这篇的基础上做了改进，打破了FastV模式固定剪枝层数和剪枝率的做法，用一个分类器去"软化"这两个参量的选取，使其更接近最优解。

- Img token注意力的衰减是一个时序过程，但不是一直单调下降的。

  采用简单的分类器剪枝的整个过程相当于一个函数：

  $$f(i_{th}\ token\ distribution) \rightarrow (i+1)_{th}\ token\ distribution$$

多层迭代：

$$f(f(\dots f(1_{th}\ token\ distribution))) \rightarrow final\ token\ distribution$$

只考虑上一轮迭代的distribution，倾向于将image token占比低的distribution反馈为image token占比更加低的distribution。

是不是可以考虑用一个RNN这种时序性的预测器，考虑过去 N 步的注意力分布向量会更好一点？