# 基于分层矩阵熵的大型视觉语言模型 Token 剪枝

林莉淇、李帅飞、吴海垚 、张云起

(中国科学技术大学　信息科学技术学院)

## 摘　要

近年来，大型视觉语言模型（LVLM）在处理图文多模态任务时展现出卓越的能力，但其巨大的计算开销严重制约了应用效率。这主要源于模型在处理大量视觉 Token 时，自注意力机制所带来的二次方计算复杂度。为解决此问题，现有方法多采用基于注意力分数的 Token 剪枝策略，但该类方法依赖于启发式假设，未必能最精确地衡量 Token 的固有信息量。

本文旨在从信息论第一性原理出发，为 LVLM 的 Token 剪枝提供一个更鲁棒的理论框架。我们提出了一种基于矩阵熵（Matrix-based Entropy）的 Token 剪枝方案 Entropy-Prune。该方法不再依赖注意力权重，而是将 Token 的重要性直接定义为其从序列中移除后，导致的整个序列信息总量的减少量。针对朴素实现方案高达 $O(N^4)$ 的计算复杂度问题，我们设计了一种高效分层剪枝（Efficient Hierarchical Pruning）算法：通过 K-Means 将 Token 聚类，在计算成本可接受的簇的层面上评估其信息重要性，从而实现高效剪枝。此外，为了增强模型对深层特征的感知能力，我们进一步引入 Attention Rollout 机制来递归式地合并多层矩阵熵，从而获得对 Token 全局重要性更全面的评估。

**关键词**：大型视觉语言模型；Token 剪枝；矩阵熵；信息论

# Abstract

In recent years, Large Vision-Language Model (LVLM) have demonstrated remarkable capabilities in handling multimodal tasks, yet their significant computational overhead severely limits practical efficiency. This challenge primarily stems from the quadratic computational complexity of the self-attention mechanism when processing a large number of visual Tokens. To address this, existing methods often employ Token pruning strategies based on attention scores. However, such approaches rely on heuristics and may not accurately measure the intrinsic information content of each Token.

This paper aims to provide a more robust theoretical framework for Token pruning in LVLM, grounded in the first principles of information theory. We propose EntropyPrune, a Token pruning scheme based on Matrix-based Entropy. This method moves beyond reliance on attention weights and instead defines a Token's importance as the reduction in the total information content of the sequence upon its removal. To overcome the prohibitive $O(N^4)$computational complexity of a naïve implementation, we design an Efficient Hierarchical Pruning algorithm. This algorithm first clusters Tokens using K-Means and then assesses their information importance at the computationally tractable cluster level, enabling efficient pruning. Furthermore, to enhance the model's awareness of features across different depths, we introduce the Attention Rollout mechanism to recursively fuse matrix entropies from multiple layers, yielding a more comprehensive assessment of a Token's global importance.



**Keywords:** Large Vision-Language Model (LVLM); Token Pruning; Matrix Entropy; Information Theory

# 第一章　引言

近年来，大型视觉语言模型 (Large Vision-Language Model, LVLM) 在视觉问答、图像描述等领域取得了突破性进展，展现出强大的图文理解能力。这些模型通常将图像编码为大量的视觉 Token，并与文本 Token 一同送入大型语言模型进行处理。然而，这种方法的成功也带来了巨大的计算挑战：自注意力机制的计算复杂度与输入 Token 序列的长度成平方关系 $O(N^2)$，海量的视觉 Token 使得模型在推理时对计算资源和内存的消耗急剧增加，严重制约了其在处理高分辨率图像和实际部署中的应用效率。

为了应对这一挑战，学术界提出了多种 Token 剪枝策略，旨在识别并剔除信息冗余的视觉 Token。其中，以 FastV[1] 为代表的、基于注意力分数的方法是当前的主流。这类方法通过分析模型内部的注意力权重，保留对最终输出贡献最大的 Token。该方法大大减少了 LVLM 的推理时内存的消耗，提升了模型输出的速度。尽管这些方法在实践中取得了显著的效率提升，但据信息论相关知识可知，信息论中拥有能更好评估 Token 内在包含的、不可替代的信息量的方法，因此，本文探索一种更为根本、源于信息论第一性原理的重要性度量标准，有望为 Token 剪枝提供一个更鲁棒、更精确的理论基础。

本文旨在突破现有方法的局限，提出一种全新的、基于矩阵熵（Matrix-based Entropy）的 Token 剪枝框架。我们不再依赖模型内部的注意力分布，而是直接从信息论的角度出发，将 Token 的重要性定义为其从序列中移除后，整个序列信息总量（即冯 · 诺依曼熵）的减少量。这一方法能够更精确地量化每个 Token 的边际信息贡献。然而，直接通过“留一法”计算每个 Token 重要性的朴素方案会导致高达 $O(N^4)$ 的计算复杂度，使其不具备实用性。为此，我们进一步设计了一种高效分层剪枝（Efficient Hierarchical Pruning）算法：首先通过 K-Means 将 Token 高效地聚类为若干语义簇，然后在计算成本可接受的簇的层面上应用矩阵熵进行重要性评估，最后仅从最重要的簇中选取代表性 Token 予以保留。

此外，为了克服单层特征可能存在的局限性，我们进一步引入了 Attention Rollout 机制来对多层的矩阵熵进行合并。通过递归式地融合模型浅层到深层计算出的矩阵熵，我们的方法能够捕捉信息在网络深度维度上的流动与演变，从而获得对 Token 全局重要性的更深度、更全面的感知能力。

本报告的组织结构如下：第二章详细介绍基于矩阵熵的 Token 剪枝方案及其分层优化策略以及相关算法，第三章介绍并探如何利用 Attention Rollout 机制进行多层信息融合。第四章将介绍我们的实验设置, 实验结果以及性能比较。第五章对全文工作进行总结与展望。最后，附录部分提供了相关算法的实现细节。

# 第二章　基于矩阵熵的 Token 剪枝

## 2.1　矩阵熵的研究背景

在人工智能的前沿探索中，多模态表示学习已成为构建能够理解并与复杂世界交互的智能体的核心基石。从视觉问答（VQA）、图像字幕生成到跨模态检索，这些任务的成功无不依赖于模型将来自不同感知渠道（如视觉和语言）的信息，映射到一个统一且语义一致的共享表示空间的能力。实现这种有效的语义对齐，关键在于最大化不同模态间配对数据（例如，一张图片及其对应的文本描述）的互信息。互信息作为一种源于信息论的无监督度量标准，能够量化两个变量之间的依赖关系，使其成为评估和优化跨模态表示质量的理想选择。

为了实现这一目标，传统的研究方法主要沿着两条技术路径演进。第一条路径是经典的非参数估计方法，例如基于数据分箱或核密度估计。这些方法在理论上非常直观，它们试图先估计出数据在高维空间中的概率密度函数，再代入香农公式计算互信息。然而，这些方法在实践中面临着严峻的“维度灾难”问题。对于深度学习模型产生的数百乃至数千维的特征向量，准确估计其概率密度所需的样本量呈指数级增长，使得这类方法在现代 AI 应用中几乎不具备可行性。为了克服高维数据的挑战，第二条路径——基于神经网络的变分下界估计方法应运而生，其中以 Wang et al.[2] 提出的互信息神经估计器（MINE）最为知名。MINE 通过引入一个辅助的判别器网络，将复杂的 MI 计算问题巧妙地转化为一个更易于处理的、可微的下界优化问题。尽管 MINE 及其变体在许多任务中取得了显著成功，但它们也引入了新的复杂性。该框架的性能高度依赖于辅助判别器的设计与训练，这不仅增加了模型整体的复杂度，还可能引入对抗性训练过程中的不稳定性。此外，优化一个可能不紧的（non-tight）下界，也无法保证总能精确地导向互信息的最大化。因此，学术界和工业界持续寻求一种既能有效处理高维数据，又具备更高稳定性和实现简洁性的互信息优化新范式。

本研究引入并应用了一种非对抗性的互信息最大化框架——矩阵熵差异最大化（Difference of Matrix-based Entropies, DiME）。DiME 方法[3] 从一个全新的视角切入，它彻底摒弃了对概率密度的直接估计和对辅助判别器网络的依赖。其核心思想根植于谱图理论，通过计算样本特征向量所构成的格拉姆 Gram 矩阵的冯 · 诺依曼熵来直接量化数据集的信息量。

## 2.2　FastV 的 Token 剪枝方法

在多模态大模型（如 LLaVA[4]）中，图像通常被编码成大量的视觉 Token，这些 Token 随后与文本 Token 一起输入到语言模型中进行处理。虽然这为模型提供了丰富的视觉信息，但大量的视觉 Token 会显著增加推理的计算复杂度和内存消耗，尤其是在处理高分辨率图像或批处理时。Chen et al. 提出了一种新颖的 Token 剪枝方法 FastV[1]，旨在识别并移除对模型输出贡献较小的冗余视觉 Token，从而在保持性能的同时，大幅提升推理效率。

在处理图像时，图像经常会被分割成小块，并通过视觉编码器转化为一系列视觉 Token，因此视觉 Token 随着照片的分辨率呈平方级增长。又因为在多模态大模型中，语言模型需要对所有输入的 Token 进行自注意力计算。自注意力的计算复杂度与序列长度的平方成正比 $O(N^2)$，大量的视觉 Token 使得 N 非常大，导致推理实现和内存消耗急剧增加。

但经过实验发现，图像 Token 存在注意力衰减现象：在生成文本的早期阶段，模型需要大量的参考图像信息来理解整体场景并确定回答的方向，并将其与初始文本上下文进行深度融合，因此在初期视觉 Token 的注意力分数普遍较高。但在进入深层网络 (如第二层以后) 图像 Token 获得的平均注意力机制得分会急剧下降，仅为 System Prompts 得分的 0.21% 左右, 如图1所示。这表明一旦视觉信息在早期层被有效处理并内化到模型的隐状态中，原始的、大量的视觉 Token 对后续深层计算的直接贡献变得微乎其微。

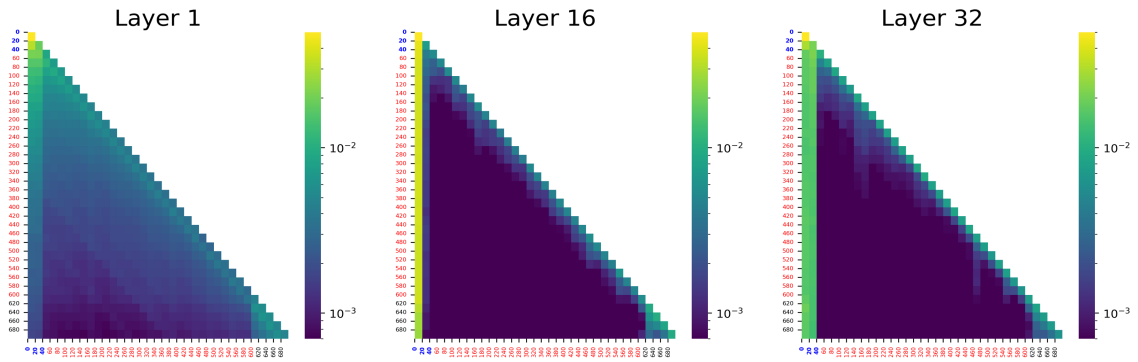


图 1: 在输入图片为 https://github.com/pkunlp-icler/FastV/blob/main/figs/example.jpg，Prompt 为 Describe all the objects in the image 时,LLAVA 模型[4] 对图片 Token 在不同深度层的注意力分数变化

因此，FastV 提出了 Top-K 剪枝策略：

- 非剪枝阶段：从模型的第 0 层到第 FastV_k-1 层（即前 k 层），所有视觉 Token 都将保持完整，不进行任何剪枝操作。这确保了模型在处理视觉信息的关键早期阶段，能够充分利用所有图像细节，完成视觉特征的提取和与文本的初始融合。这与上述“早期层高注意力得分”的观察相符，保护了模型理解视觉信息的关键环节。

- 剪枝阶段：从模型的第 FastV_k 层开始，FastV 的动态剪枝策略将被激活。它会选择注意力分数最高的视觉 Token 的前 R% 进行保留，而其余的视觉 Token 则被剪枝（即在后续的自注意力计算中被忽略）。

通过基于图像 Token 注意力分数的 Token 剪枝，有效减少语言模型深层中参与计算的视觉 Token 数量，大幅降低了自注意力机制的计算开销；同时也减少输入序列的有效长度直接减少了 GPU 内存中需要存储的 Token 隐状态和中间激活，使得模型能够处理更长的序列或在内存受限的环境下运行；并且 FastV 是一种纯粹的推理优化技术，可以直接应用于已预训练好的 LLaVA 模型，无需进行额外的训练或微调，极大地简化了部署流程。

## 2.3 基于矩阵熵的 Token 剪枝方案

为了降低大语言模型在处理冗长序列时的计算开销，同时最大限度地保留序列的语义信息，DIME 提出一种基于矩阵熵的动态 Token 剪枝方法。该方法的核心思想是，一个 Token 的重要性取决于其对整个序列信息总量的贡献度。通过量化每个 Token 的边际信息贡献，我们可以识别并剔除冗余的 Token，从而在不显著影响模型性能的前提下，提升推理效率。

DIME 的方法建立在矩阵的冯 · 诺依曼熵之上，它通过分析数据协方差矩阵的谱来量化信息量，从而避免了在高维空间中直接估计概率密度的难题。给定一个由 $N$ 个 Token 嵌入向量组成的序列 $\mathcal{T} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_N\}$，其中每个向量 $\boldsymbol{t}_i \in \mathbb{R}^D$。首先，我们对每个向量进行 L2 归一化，得到 $\hat{\boldsymbol{t}}_i = \boldsymbol{t}_i / \|\boldsymbol{t}_i\|_2$。然后，我们构建格拉姆矩阵（Gram matrix）$\mathrm{K} \in \mathbb{R}^{N \times N}$，其元素定义为：

$$K_{ij} = \hat{\boldsymbol{t}}_i^\top \hat{\boldsymbol{t}}_j \tag{1}$$

矩阵 K 捕捉了序列中所有 Token 对之间的余弦相似度。该矩阵的特征值 $\{\lambda_j\}_{j=1}^N$ 反映了数据在特征空间中的主要变化方向和能量分布。我们将这些特征值归一化，使其总和为 1，得到伪概率分布 $\{p_j\}_{j=1}^N$，其中 $p_j = \lambda_j / \sum_{k=1}^N \lambda_k$。

序列 $\mathcal{T}$ 的矩阵熵 $H(\mathcal{T})$ 定义为：

$$H(\mathcal{T}) = -\sum_{j=1}^{N} p_j \log(p_j + \epsilon) \tag{2}$$

其中 $\epsilon$ 是一个为了防止数值计算不稳定而加入的极小正常数。

基于此，我们定义第 $i$ 个 Token $\boldsymbol{t}_i$ 的重要性得分 $\mathcal{S}_i$ 为：从序列中移除该 Token 所导致的整体熵减量。令 $\mathcal{T}_i = \mathcal{T} \setminus \{\boldsymbol{t}_i\}$ 表示移除了 $\boldsymbol{t}_i$ 的子序列，则其重要性得分为：

$$\mathcal{S}_i = H(\mathcal{T}) - H(\mathcal{T}_i) \tag{3}$$

直观上，如果一个 Token 包含独特的信息，移除它将导致整体熵大幅下降，其得分 $\mathcal{S}_i$ 会很高。反之，如果一个 Token 是冗余的，移除它对整体熵影响甚微，其得分则会很低。

根据公式 (3)，最直接的实现方式（我们称之为“朴素方案”）是遍历每一个 Token，通过“留一法”（Leave-One-Out）计算其重要性得分，然后保留得分最高的 $k$ 个 Token。

然而，该朴素方案的计算复杂度约为 $O(N^4)$，其中 $N$ 是序列长度。这是因为需要进行 $N$ 次子序列的熵计算，而每次熵计算本身涉及到复杂度为 $O(N^3)$ 的特征值分解。当处理现代视觉模型（如 ViT）产生的长序列（$N$ 通常为 256 或更多）时，如此高的计算成本使其在实际应用中是不可行的。

为了解决朴素方案的计算瓶颈，我们设计了一种高效分层剪枝（Efficient Hierarchical Pruning）方案。该策略的核心是将计算成本高昂的矩阵熵评估嵌套在一个“分而治之”的框架内，使其作用于更高层级的语义单元（簇），而非单个 Token[5]。输入一个包含 $N$

个 Token 嵌入的原始长序列 $\mathcal{T}$。我们采用 K-Means 算法，一种计算高效且广泛应用的聚类方法，将 $N$ 个高维 Token 向量划分为 $M$ 个语义簇 $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M\}$，其中簇的数量 $M \ll N$（例如，N=256, M=32）。此步骤将语义上邻近的 Token 自然地聚合在一起。为了在宏观层面进行评估，我们为每个簇计算一个代表其核心语义的向量。我们采用簇质心（Centroid）作为其代表，即簇内所有 Token 向量的平均值：

$$\boldsymbol{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{t}_i \in \mathcal{C}_j} \boldsymbol{t}_i \tag{4}$$

由此，我们获得了一个由 $M$ 个质心向量构成的“元序列” $\mathcal{T}_{\text{cluster}} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_M\}$。我们将矩阵熵评估理论应用于这个长度为 $M$ 的元序列 $\mathcal{T}_{\text{cluster}}$ 上。通过“留一法”，我们计算出移除每一个簇代表后，元序列整体熵的减损量，以此作为该簇的重要性得分 $\mathcal{S}_{\text{cluster}_j}$：

$$\mathcal{S}_{\text{cluster}_j} = H(\mathcal{T}_{\text{cluster}}) - H(\mathcal{T}_{\text{cluster}} \setminus \{\boldsymbol{c}_j\}) \tag{5}$$

由于 $M$ 的数值远小于 $N$，此步骤的计算是高效且可行的。在获得所有簇的重要性得分后，我们执行最终的剪枝操作。

1. **筛选核心簇：** 根据得分对 $M$ 个簇进行排序，并选取得分最高的 $k_c$ 个“核心簇”。
2. **选择代表性 Token：** 遍历这 $k_c$ 个核心簇，并从每个簇 $\mathcal{C}_j$ 内部，选取 $t_{p_c}$ 个最具代表性的 Token（例如，选择离该簇质心最近的 Token）。
3. **自适应数量调整：** 在此步骤中，我们引入了自适应机制。若某个被选中的核心簇 $\mathcal{C}_j$ 的成员数量小于 $t_{p_c}$，算法将不会为了凑数而选择其他 Token，而是稳健地保留该簇内的所有成员。因此，最终保留的 Token 总数是一个小于等于 $k_c \times t_{p_c}$ 的动态值。这种设计确保了每一个被保留的 Token 都严格来自于高重要性的语义簇，并且是其所在簇的优质代表，从而避免了引入次级信息。


图 2: 在输入图像为 256Tokens 时,K-Means 聚类多簇结果可视化显示

算法 1 详细描述了上述高效分层剪枝方案的完整流程。

**Algorithm 1** 高效分层 Token 剪枝算法 (Hierarchical Pruning Algorithm)

**Require:** 原始 Token 嵌入序列 $\mathcal{T} = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_N\}$
目标簇数量 $M$
要保留的核心簇数量 $k_c$
每个核心簇保留的代表性 Token 数 $t_{p_c}$
**Ensure:** 剪枝后的 Token 序列 $\mathcal{T}_{\text{pruned}}$

{— 步骤一: 将 Token 聚类成 M 个语义簇 —}
$\{\mathcal{C}_1, \ldots, \mathcal{C}_M\} \leftarrow \text{KMeans}(\mathcal{T}, M)$ {$\mathcal{C}_j$ 是一个包含 Token 向量的集合}
{— 步骤二: 生成簇代表的元序列 —}
$\mathcal{T}_{\text{cluster}} \leftarrow \emptyset$
**for** $j = 1$ **to** $M$ **do**
  $\boldsymbol{c}_j \leftarrow \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{t} \in \mathcal{C}_j} \boldsymbol{t}$ {计算第 j 个簇的质心}
  $\mathcal{T}_{\text{cluster}} \leftarrow \mathcal{T}_{\text{cluster}} \cup \{\boldsymbol{c}_j\}$ {构建质心序列}
**end for**
{— 步骤三: 评估每个簇的重要性 —}
$\{\mathcal{S}_1, \ldots, \mathcal{S}_M\} \leftarrow \text{ComputeClusterImportance}(\mathcal{T}_{\text{cluster}})$ {嵌套矩阵熵评估}
{— 步骤四: 执行自适应分层剪枝 —}
$\mathcal{I}_{\text{top\_clusters}} \leftarrow \text{TopK\_Indices}(\{\mathcal{S}_j\}_{j=1}^{M}, k_c)$ {获取得分最高的簇索引}
$\mathcal{T}_{\text{pruned}} \leftarrow \emptyset$ {初始化空的剪枝后集合}
**for** each $j \in \mathcal{I}_{\text{top\_clusters}}$ **do**
  {遍历所有被选中的高分簇, 并从中选择最具代表性的 Token}
  $\mathcal{T}_{\text{repr}} \leftarrow \text{FindNearestTokens}(\mathcal{C}_j, \boldsymbol{c}_j, t_{p_c})$ {查找离质心最近的 $t_{p_c}$ 个 Token}
  $\mathcal{T}_{\text{pruned}} \leftarrow \mathcal{T}_{\text{pruned}} \cup \mathcal{T}_{\text{repr}}$
**end for**
Sort $\mathcal{T}_{\text{pruned}}$ by original Token indices (optional) {按原始位置排序}
**return** $\mathcal{T}_{\text{pruned}}$

通过执行算法 1，我们可以得到一个信息密度更高、长度更短的 Token 序列。该序列随后可被送入模型的后续层级进行处理，从而实现计算优化的目标。

# 第三章　基于 Attention Rollout 机制对矩阵熵进行合并

Attention Rollout 机制[6] 是一种在大型视觉语言模型 (LVLM) 中合并多层注意力以作为 Token 剪枝时的评分标准的机制, 该机制可以捕捉特征在深度网络中的长程依赖, 避免了过度依赖某一层输出的问题

这里利用 Attention Rollout 机制对上文中计算的矩阵熵进行合并, 可以增强特征重要性评估的深度感知能力整合 $l_1$ 到 $l_2$ 层的矩阵熵, 我们有下面的公式

$$\tilde{\mathrm{H}}^{l}=\begin{cases}\mathrm{H}^{l_1}, & \text{if } l=l_1,\\ (\mathrm{H}^{l}+\mathrm{I})\tilde{\mathrm{H}}^{l-1}, & \text{if } l_1<l\leq l_2.\end{cases} \tag{6}$$

通过 Attention Rollout 机制递归地合并每一层的矩阵熵, 可以更准确地评估全局信息的信息量和 Token 之间的互信息大小，具体实现流程图如图3

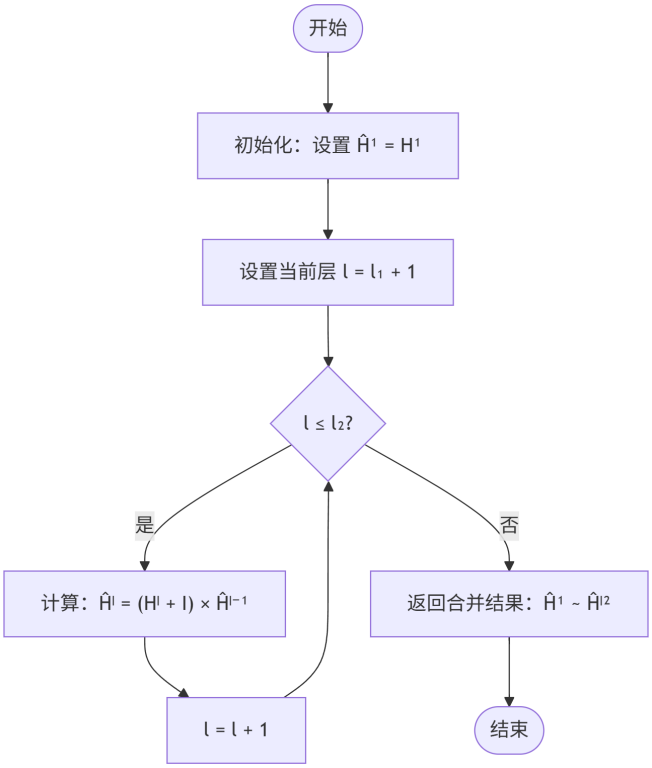


图 3: Attention Rollout 合并矩阵熵

# 第四章　实验

在本章节中，我们设计了一系列实验来评估以下两种 Token 剪枝策略：基于矩阵熵的剪枝（Entropy-Prune）和基于 FastViT 的剪枝（FastViT-Prune），选取了三种经典的基线模型进行对照，旨在研究新剪枝方案的性能与效率优化策略的可行性。

## 4.1　实验设置

### 4.1.1　数据集

我们所有的实验均在经典的 `COCO 2017` 数据集上进行。我们遵循标准的 “Karpathy 划分法”，将数据集划分为训练集、验证集和测试集。

### 4.1.2　评估指标

我们的评估分为两个维度：

- **描述生成质量指标**: 我们采用图像描述领域全套的标准化评估指标，包括 BLEU-4, METEOR, ROUGE-L, CIDEr, 和 SPICE。其中，CIDEr 和 SPICE 是衡量描述与人类参考描述在语义上一致性的关键指标。
- **模型效率指标**:
  - **FLOPs (G)**: 衡量视觉编码器处理 Token 所需的总计算量（十亿浮点运算次数）。
  - **推理延迟 (Latency, ms)**: 在单张 RTX 3090 GPU 上，测量处理单个样本所需的端到端时间。
  - **Token 保留率 (%)**: 剪枝后保留的视觉 Token 占原始 Token 的百分比。

### 4.1.3　对比方法

为验证我们方法的有效性，我们设置了以下几组对比：

- `LLaVA-1.5-7B`: 未经任何剪枝的原始 `LLaVA-1.5-7B` 模型，作为我们性能和效率的上限基线。
- `RandomPrune`: 随机剪枝掉 $k\%$ 的视觉 Token，作为下限基线，用于健全性检查。
- `ToMe (Token Merging)`: 一种经典且高效的 Token 合并/剪枝方法，作为我们强有力的 SOTA 基线。
- `FastViT-Prune (Ours)`: 我们实现的基于 `FastViT` 思想的 Token 剪枝策略。
- `Entropy-Prune (Ours)`: 我们提出的核心方法，即基于 Attention Rollout 合并的矩阵熵 Token 剪枝策略。

### 4.1.4 实现细节

我们所有的实验都基于 `LLaVA-1.5-7B` 模型。实验在 PyTorch2.0.0 框架下进行，硬件环境为 RTX 3090 GPU，CUDA 版本为 11.8，Python 版本 3.8（Ubuntu20.04），内存 24G。所有方法在剪枝后均不进行微调（Fine-tuning），以公平地评估剪枝策略本身带来的影响。剪枝率 $k\%$ 统一设置为 50%。

## 4.2 实验结果与分析

我们在 50% 的剪枝率下，对所有方法在 COCO 测试集上的表现进行了图像描述质量与效率的全面对比。详细的量化结果如表1所示。

表 1: 在 COCO Karpathy-test split 上，不同剪枝策略（50% 剪枝率）的性能与效率对比

| 方法 | 描述质量指标 | | | | | 效率指标 | |
|---|---|---|---|---|---|---|---|
| (Method) | BLEU-4 | METEOR | ROUGE-L | CIDEr | SPICE | FLOPs (G) | Latency (ms) |
| `LLaVA-1.5-7B` | **37.52** | **29.10** | **57.80** | 99.80 | **22.50** | 60.02 | 4221 |
| `RandomPrune` | 28.93 | 24.31 | 51.02 | 93.41 | 18.11 | 44.15 | 3418 |
| `ToMe` | 36.12 | 28.53 | 56.91 | 98.34 | 21.79 | 43.09 | 3105 |
| `FastViT-Prune` | 36.49 | 28.62 | 57.04 | **100.94** | 21.93 | **41.52** | **2815** |
| **`Entropy-Prune`** | 36.84 | 28.81 | 57.33 | 98.27 | 22.14 | 51.63 | 3472 |

从性能与效率指标对比，可以得出以下结论：FastViT-Prune 可以显著提高原模型的生成质量与效率，Entropy-Prune 方案在实际应用中会增加其计算复杂度，致使其延迟明显增大（这是不可避免地，尽管我们已经尽可能降低处理方案的复杂度），而生成质量较基线模型降低较少，如果在算力充裕的情况下，具备一定的可行性。

基于矩阵熵的 Token 剪枝前后各簇间重要性分数与相似度矩阵如图4与图5所示：

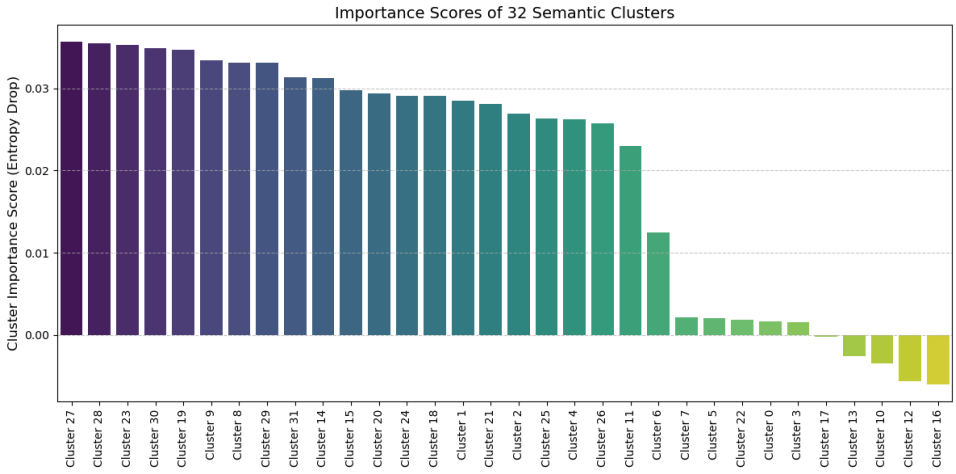


图 4: Entropy-Pruning 剪枝前后各簇重要性分数

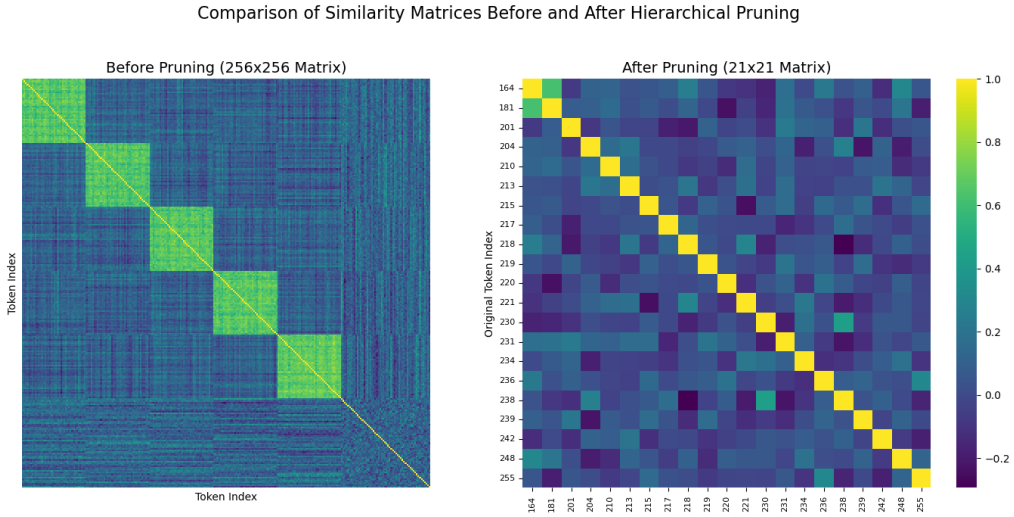


图 5: Entopy-Pruning 剪枝前后 Token 相似度矩阵

为了评估剪枝操作对数据整体结构的影响，我们对比了剪枝前后的 Token 相似度矩阵（如图5所示）。热力图的变化直观地展示了剪枝的效果：

剪枝前，一个 $256 \times 256$ 的大型矩阵清晰地暴露了原始序列的内在结构。图中存在多个明亮的非对角线“方块”，这些区域是信息冗余的直接可视化证据，表明多个 Token 在语义上高度相似，形成了功能上的重复。这张图本身就为进行剪枝提供了充分的理由。

剪枝后，矩阵的规模被急剧压缩至 $21 \times 21$，这是由于 Token 选取的自适应机制，在算法评选出的 16 个核心语义簇中，有 11 个为仅包含单个 Token 的‘独特点’簇，保证了最终剪枝后的序列是对原始核心信息更高保真度的提炼，这直接对应了后续计算量与内存占用的数量级下降。更重要的是，矩阵的结构发生了根本性的变化。原先代表冗余的亮色方块结构已完全消失，取而代之的是一个更加稀疏、接近准对角化的矩阵。这表明，剪枝操作移除了每个语义簇内部的冗余成员，只保留了能够唯一代表该簇的核心 Token。最终得到的序列，其成员彼此间的语义相似度较低，构成了一组近似正交的“语义基向量”。

# 第五章　总结与展望

本文从信息论的角度研究了多种针对大型视觉语言模型 (LVLM) 的 Token 剪枝方法, 包括 FastV 和基于矩阵熵的剪枝方案。FastV 通过注意力分数进行剪枝, 而基于矩阵熵的剪枝方法利用矩阵熵来量化信息量, 从而剪掉信息量较少的 Token。最后通过基于 Attention Rollout 机制的矩阵熵合并, 来评估剪枝后每一层的信息量及 Token 之间的互信息大小。

尽管我们的研究方案在模型性能优化上取得了一定的进展, 但是还存在很大的改进空间:

- 虽然基于矩阵熵的 Token 剪枝方法对模型推理速度有提升, 但是相较于其他 Token 剪枝方法而言, 该方法在计算矩阵熵时有较大的时间开销, 需要进行优化。
- 当前方案簇的个数 $M$, 保留簇的数量 $k_c$ 作为一个超参数设定, 未来的研究需要让模型根据输入内容的复杂度, 动态地, 自适应地调整超参数。

此外信息论中的其他理论如 Zhang et al.[7] 提出的利用渐近均分性通过分析文本和布局等不同模态 Token 之间的相关性，来智能地移除冗余信息；Chen et al.[8] 提出的基于率失真理论实现视觉上下文压缩方法；Cheng et al.[9] 提出的使用最优传输理论将 Token 重要性评估转化为最小化传输代价的优化问题等，这些方法在大型视觉语言模型 (LVLM) 的 Token 剪枝中仍有广阔的应用前景, 这都将是我们未来工作的一个方向与重点。

# 第六章　附录

## 6.1　FastV 代码实现

### 6.1.1　方法一：原地 Token 剪枝

基于“文本对图像的注意力”来识别并保留最重要的视觉 Token，同时丢弃次要的视觉 Token，从而高效地压缩了序列长度，达到加速模型推理的目的。

**Algorithm 2** FastV 原地剪枝算法 (FastVInplacePruning)

**Require:** 当前隐藏状态 $H$, 上一层的注意力分数 $A$, 当前层索引 $idx$, 关键决策层索引 $L_{agg}$, 保留的视觉 Token 数量 $K$, 系统提示 Token 长度 $N_{sys}$, 视觉 Token 总长度 $N_{img}$

**Ensure:** 剪枝后的隐藏状态 $H'$, 更新后的位置 ID $P'$, 更新后的注意力掩码 $M'$

**if** $idx = L_{agg}$ **then**
  $A_{avg} \leftarrow$ AverageScoresAcrossHeads$(A)$ {在所有注意力头之间平均分数}
  $S_{text \to all} \leftarrow$ SelectScoresFromLastToken$(A_{avg})$ {提取由文本 Token 产生的注意力}
  $S_{img} \leftarrow$ ExtractImageTokenScores$(S_{text \to all}, N_{sys}, N_{img})$ {分离出对视觉 Token 的注意力分数}
  $I_{\text{top\_k}} \leftarrow$ GetIndicesOfTopKScores$(S_{img}, K)$ {找到最重要的 K 个视觉 Token 的索引}
  $I_{\text{top\_k}} \leftarrow$ AdjustIndicesToGlobalScope$(I_{\text{top_k}}, N_{sys})$ {将局部索引映射回全局序列索引}
  $I_{sys} \leftarrow$ GetIndicesOfSystemTokens()
  $I_{text} \leftarrow$ GetIndicesOfTextTokens()
  $I_{\text{keep}} \leftarrow$ Concatenate$(I_{sys}, I_{\text{top_k}}, I_{text})$ {合并所有需要保留的 Token 索引} $I_{\text{keep}} \leftarrow$ Sort$(I_{\text{keep}})$ {排序以维持原始顺序}
  $H' \leftarrow$ FilterTensorByIndex$(H, I_{\text{keep}})$ {物理剪枝：根据索引过滤隐藏状态}
  $P' \leftarrow I_{\text{keep}}$ {更新位置 ID}
  $M' \leftarrow$ GenerateNewMaskFor$(H')$ {为新序列生成注意力掩码}
  **return** $H', P', M'$
**else**
  **return** 未经修改的输入 ($H$, $P$, $M$)
**end if**

### 6.1.2　方法二：注意力掩码

此方法不改变序列的物理长度，而是通过修改注意力掩码来阻止模型“看到”不重要的 Token。

**Algorithm 3** FastV 注意力掩码剪枝算法

**Require:** 当前隐藏状态 $H$, 当前层索引 $idx$, 关键决策层索引 $L_{agg}$, 上一层的注意力分数 $A_{last}$, 保留的视觉 Token 数量 $K$, 以及各部分 Token 的长度与索引

**Ensure:** 为当前层计算出的新注意力掩码 $M'$

**if** $idx < L_{agg}$ **then**
  $M' \leftarrow$ CreateDefaultFullAttentionMask()
  {在决策层之前，所有 Token 均可见。}
**else if** $idx = L_{agg}$ **then**
  {在关键决策层，生成一次性的剪枝掩码。}
  **if** $idx > 0$ **then**
    {常规情况：基于上一层注意力进行决策}
    $S_{img} \leftarrow$ CalculateImageTokenImportanceScores$(A_{last})$
    $I_{\text{top\_k}} \leftarrow$ GetIndicesOfTopKScores$(S_{img}, K)$
    $M_{new} \leftarrow$ CreateBooleanMaskWithAllVisible()
    SetVisibility$(M_{new}, \text{all_visual_Tokens}, \text{False})$ {先隐藏所有视觉 Token}
    SetVisibility$(M_{new}, I_{\text{top_k}}, \text{True})$ {再重新激活重要的 Token}
    $M' \leftarrow$ FormatMaskForDecoder$(M_{new})$
    CachePrunedMask$(M')$ {缓存此掩码以供后续层使用}
  **else**
    {边缘情况：在第一层，无历史注意力，随机选择}
    $I_{\text{random}} \leftarrow$ GetKRandomIndices$(\text{all_visual_Tokens}, K)$
    $M_{new} \leftarrow$ CreateBooleanMaskWithAllVisible()
    SetVisibility$(M_{new}, \text{all_visual_Tokens}, \text{False})$
    SetVisibility$(M_{new}, I_{\text{random}}, \text{True})$
    $M' \leftarrow$ FormatMaskForDecoder$(M_{new})$
    CachePrunedMask$(M')$ {缓存此掩码以供后续层使用}
  **end if**
**else**
  {即 $idx > L_{agg}$}
  $M' \leftarrow$ GetCachedPrunedMask()
  {在决策层之后，持续使用已生成的剪枝掩码。}
**end if**
**return** $M'$

## 6.2 Attention Rollout 机制

Attention Rollout 机制对上文中计算的矩阵熵进行合并, 增强特征重要性评估的深度感知能力

**Algorithm 4** 基于 Attention Rollout 的矩阵熵合并

**Require:** $l_1 \sim l_2$ 层中每一层计算出的矩阵熵 $\mathrm{H}^l$

**Ensure:** $l_1 \sim l_2$ 层中每一层合并后的矩阵熵 $\widehat{\mathrm{H}}^l$

{— 步骤一: 初始化 —}

$\widehat{\mathrm{H}}^{\mathrm{l}_1} = \mathrm{H}^{\mathrm{l}_1}$

{— 步骤二: 递归地合并每一层的矩阵熵 —}

**for** $l = l_1 + 1$ **to** $l_2$ **do**

$\widehat{\mathrm{H}}^{\mathrm{l}} = (\mathrm{H}^{\mathrm{l}} + \mathrm{I})\widehat{\mathrm{H}}^{\mathrm{l}-1}$

**end for**

**return** $\widehat{\mathrm{H}}^{\mathrm{l}_1} \sim \widehat{\mathrm{H}}^{\mathrm{l}_2}$

# 参考文献

[1] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang, "An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models," 2024.

[2] B. Wang, S. Wang, Y. Cheng, Z. Gan, R. Jia, B. Li, and J. Liu, "Infobert: Improving robustness of language models from an information theoretic perspective," *ArXiv*, vol. abs/2010.02329, 2020.

[3] O. Skean, J. K. Hoyos-Osorio, A. J. Brockmeier, and L. G. S. Giraldo, "Dime: Maximizing mutual information by a difference of matrix-based entropies," *ArXiv*, vol. abs/2301.08164, 2023.

[4] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[5] X. Wu, F. Zeng, X. Wang, and X. Chen, "Ppt: Token pruning and pooling for efficient vision transformers," *arXiv preprint arXiv:2310.01812*, 2023.

[6] H. Chen, Y. Ni, W. Huang, Y. Liu, S. Jeong, F. Wen, N. Bastian, H. Latapie, and M. Imani, "Vltp: Vision-language guided token pruning for task-oriented segmentation," *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 9353–9363, 2024.

[7] R. Zhang, Y. Lyu, R. Shao, G. Chen, W. Guan, and L. Nie, "Token-level correlation-guided compression for efficient multimodal document understanding," *ArXiv*, vol. abs/2407.14439, 2024.

[8] J. Chen, L. Ye, J. He, Z. Wang, D. Khashabi, and A. L. Yuille, "Efficient large multi-modal models via visual context compression," in *Neural Information Processing Systems*, 2024.

[9] C. Yang, Y. Sui, J. Xiao, L. Huang, Y. Gong, C. Li, J. Yan, Y. Bai, P. Sadayappan, X. Hu, and B. Yuan, "Topv: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model," *ArXiv*, vol. abs/2503.18278, 2025.