# 多模态RAG场景的表征压缩比较研究

## 摘要

本研究在多模态检索增强生成（RAG）及后续问答场景，针对高分辨率图片数量多导致的存储开销过大的问题，提出表征压缩的思路，对比了主成分分析（PCA）、典型相关分析（CCA）和向量量化（VQ）等方法；其中 PCA 通过正交变换保留数据最大方差实现线性降维，CCA 分析图像与文本特征的线性相关性以寻找最大典型相关变量，VQ 则将高维向量映射到低维码本（如 int8、int4 量化）。实验方面在 Flickr8k 数据集上利用 CLIP 和 BLIP 模型实验发现，4 位向量量化在节省 87.5% 存储下效果仅降 36.65% 但测试耗时增加，CCA 节省 50% 存储且准确率下降最少（35.65%）但需要训练耗时，PCA 因未考虑图文内积交互效果最差。同时指出 CLIP 对比学习思想源于典型相关分析，二者区别在于前者通过梯度下降更新参数并利用负样本约束，后者通过特征值求解投影矩阵解析解。本项目的代码链接为https://github.com/LIU-Hao-2002/Multimodal_RAG_VQA_Project.git

## 问题背景

1. 场景定义：对于用户含有大量图片的手机相册库，有时候查找照片很麻烦。现有的相册往往按照时间排序，对于特定一张照片可能因为忘记日期而搜索不到，仅仅记得图片上有什么内容，例如“一张含有猫的图片”。
2. 问题定义：多模态RAG研究的问题之一就是如何基于文本搜索图片（多模态搜索），然后基于图片做问答（多模态理解）。例如用户模糊描述图片“a picture of a cat”搜索到图片后，大模型基于这张图片回答用户的多个问题
3. 问题卡点：当图片分辨率很高且数量很多时，在手机内为每张图片生成描述（caption）并存储每张图片的图文双表征并不现实。一方面因为caption信息（grounding）往往不够精确导致搜索结果不佳，另一方面主要是手机作为端侧设备存储空间有限，生成caption会增加额外的存储开销。如果仅存储高分辨率的图片表征在图片量过大时也会需要大量的存储空间。在检索向量并不需要过于精确的情况下可以进行向量压缩节省空间。
4. 研究目的：综上所述，本研究聚焦在低维空间下进行表征压缩，实现在不大幅影响检索正确率的情况下降低存储开销。

## 理论方法

向量的存储压缩分为统计降维和向量量化等方法，本研究对比三种：主成分分析，典型相关分析和向量量化方法。

1. 主成分分析：主成分分析旨在通过正交变换，将高维向量投影到低维空间，同时尽可能保留数据的最大方差，以此实现线性降维。通过这种方式，能在减少数据维度的同时，最大程度保留数据中的关键信息，从而降低后续计算和存储的复杂度。

   a. 数据预处理：假设原始数据集为 $\mathbf{X} \in \mathbb{R}^{n \times d}$ ，其中 $n$ 表示样本数量， $d$ 表示特征维数。首先对数据进行均值中心化处理，具体操作如下： $\mathbf{X}' = \mathbf{X} - \mathbf{1}_n \mu^T, \quad \mu = \frac{1}{n} \mathbf{X}^T \mathbf{1}_n$ 。这里 $\mathbf{1}_n$ 是全 1 列向量， $\mu$ 为各特征的均值。经过均值中心化，数据的分布将以原点为中心，便于后续的计算和分析。

   b. 协方差矩阵：对中心化后的数据计算协方差矩阵 $\mathbf{\Sigma} = \frac{1}{n-1} \mathbf{X}'^T \mathbf{X}' \in \mathbb{R}^{d \times d}$ 。协方差矩阵 $\mathbf{\Sigma}$ 描述了数据中各个特征之间的线性相关性，其对角线上的元素为各特征的方差，非对角线上的元素为特征之间的协方差。

   c. 特征值分解：对协方差矩阵 $\mathbf{\Sigma}$ 进行特征值分解： $\mathbf{\Sigma} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$ ；其中 $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d]$ 是正交特征向量矩阵， $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$ 是降序排列的特征值矩阵。特征值和特征向量将协方差矩阵分解为多个相互正交的方向，每个方向上的特征值表示该方向上数据的方差大小。

   d. 主成分投影：选取前 $k$ 个主成分（对应最大的k个特征值），构建投影矩阵 $\mathbf{W} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k] \in \mathbb{R}^{d \times k}$ ，将原始数据进行降维： $\mathbf{Z} = \mathbf{X}'\mathbf{W} \in \mathbb{R}^{n \times k}$ ；降维后的数据 $\mathbf{Z}$ 在保留了原始数据大部分方差（即信息）的同时，维度从 $d$ 维降低到了 $k$ 维。

   e. 累计方差贡献率：为了衡量降维后保留的信息量，引入累计方差贡献率的概念： $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$ ；该指标反映了前 $k$ 个主成分所解释的方差占总方差的比例，通常根据实际需求设定一个阈值（如0.9或0.95），以此确定合适的k值。

2. 典型相关分析：典型相关分析主要用于分析两组变量（例如图像特征和文本特征）之间的线性相关性。通过寻找两组变量的线性组合（即典型变量），使得这些典型变量之间的相关性最大化，从而揭示两组变量之间的潜在联系，为后续的多模态检索和分析提供有效手段。

   a. 数据标准化：设 $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ 和 $\mathbf{Y} \in \mathbb{R}^{n \times d_2}$ 分别为两组变量，首先对它们进行均值中心化和标准化处理，使它们的协方差矩阵变为单位阵： $\text{Cov}(\mathbf{X}) = \mathbf{I}_{d_1}, \quad \text{Cov}(\mathbf{Y}) = \mathbf{I}_{d_2}$ 同时，计算两组变量的互协方差矩阵： $\mathbf{\Sigma}_{XY} = \frac{1}{n-1} \mathbf{X}^T \mathbf{Y} \in \mathbb{R}^{d_1 \times d_2}$ ;标准化处理可以消除不同变量之间量纲和尺度的影响，使得后续计算更加合理和准确。

   b. 典型变量构造：我们的目标是寻找投影向量 $\mathbf{w} \in \mathbb{R}^{d_1}$ 和 $\mathbf{v} \in \mathbb{R}^{d_2}$ ，构造典型变量 $u = \mathbf{w}^T \mathbf{X}$ 和 $v = \mathbf{v}^T \mathbf{Y}$ ，并使它们之间的相关系数最大： $\max_{\mathbf{w}, \mathbf{v}} \ \text{corr}(u, v) = \frac{\mathbf{w}^T \mathbf{\Sigma}_{XY} \mathbf{v}}{\sqrt{(\mathbf{w}^T \mathbf{w})(\mathbf{v}^T \mathbf{v})}}$ ；为了简化问题，通常添加约束条件 $\mathbf{w}^T \mathbf{w} = 1$ 和 $\mathbf{v}^T \mathbf{v} = 1$ ，将其转化为广义特征值问题： $\mathbf{\Sigma}_{XY} \mathbf{\Sigma}_{YY}^{-1} \mathbf{\Sigma}_{YX} \mathbf{w} = \rho^2 \mathbf{\Sigma}_{XX}^{-1} \mathbf{\Sigma}_{XY} \mathbf{\Sigma}_{YY}^{-1} \mathbf{\Sigma}_{YX} \mathbf{w}$ ;当 $\mathbf{X}$ 和 $\mathbf{Y}$ 标准化后， $\mathbf{\Sigma}XX = \mathbf{I}d_1$ ， $\mathbf{\Sigma}YY = \mathbf{I}d_2$ ，问题进一步简化为： $\mathbf{\Sigma}_{XY} \mathbf{\Sigma}_{YX} \mathbf{w} = \rho^2 \mathbf{w}, \quad \mathbf{\Sigma}_{YX} \mathbf{\Sigma}_{XY} \mathbf{v} = \rho^2 \mathbf{v}$

c. 典型相关系数与变量：求解上述广义特征值问题，得到的最大特征值 $\rho_1^2$ 对应的第一对典型变量 $(u_1, v_1)$ 具有最大的相关性，依次类推可以得到k对典型变量，且满足 $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_k \geq 0$。降维后，两组变量的表征分别为：$\mathbf{Z}_X = \mathbf{X}\mathbf{W}, \quad \mathbf{Z}_Y = \mathbf{Y}\mathbf{V}, \quad \mathbf{W} \in \mathbb{R}^{d_1 \times k}, \mathbf{V} \in \mathbb{R}^{d_2 \times k}$;这些低维表征能够有效捕捉两组变量之间的线性相关关系，为后续的多模态数据分析和检索提供基础。

3. 向量量化技术：向量量化（Vector Quantization, VQ）是一种通过将高维向量映射到低维码本实现数据压缩的技术。其核心思想是将连续的向量空间离散化，用有限个代表性向量（码本中的码字）近似表示原始向量，从而减少数据存储和计算开销。在实际应用中，量化位数越低，压缩比越高，但可能引入的量化误差也越大，需要在存储效率和数据精度之间进行权衡。

   a. int8 量化理论与实现：int8 量化将原始数据映射到 8 位整数的范围内（即 - 128 到 127）。通过计算原始数据的最大值和最小值，确定一个缩放因子（scale）和零点（zero_point），将原始浮点数转换为 int8 整数。这种量化方式能显著减少数据存储空间，同时在很多场景下对模型精度影响较小，是当前较为常用的量化方案之一。

      i. **计算缩放因子和零点**：首先找出原始 embedding 矩阵中的最大值 `max_val` 和最小值 `min_val`，通过公式 `scale = (max_val - min_val) / 255` 计算缩放因子，该因子用于将原始数据的范围映射到 0-255 区间；再通过 `zero_point = np.round(-min_val / scale)` 计算零点，其作用是将 0 映射到合适的 int8 整数。

      ii. **量化操作**：使用公式 `quantized = np.round(embedding / scale + zero_point)` 将原始浮点数 embedding 转换为 int8 整数，并通过 `np.clip` 函数确保量化后的值在 - 128 到 127 的范围内。

      iii. **返回结果**：最终返回量化后的 embedding 矩阵、缩放因子和零点，这些信息在反量化时会被用到。

      iv. **反量化过程**：反量化是量化的逆过程，将 int8 整数恢复为原始的浮点数形式。通过公式 `(quantized.astype(np.float32) - zero_point) * scale`，利用存储的缩放因子和零点，将量化后的整数还原为接近原始数据的浮点数。

   b. int4 量化理论与实现：int4 量化进一步降低了量化位数，将数据映射到 4 位整数的范围内（即 0 到 15）。由于 4 位无法直接存储负数，通常采用无符号整数表示。为了充分利用存储空间，int4 量化常将两个 4 位整数打包到一个 8 位整数中，进一步提高压缩比，但也使得量化和反量化过程更为复杂。

      i. **计算缩放因子和零点**：与 int8 量化类似，先找出原始数据的最大值和最小值，通过公式 `scale = (max_val - min_val) / 15` 计算缩放因子，`zero_point = np.round(-min_val / scale).astype(np.uint8)` 计算零点。

      ii. **量化操作**：使用公式 `quantized = np.round(embedding / scale + zero_point).clip(0, 15).astype(np.uint8)` 将原始数据量化为 0 到 15 的无符号整数。

iii. **数据打包**：检查量化后数据第一个维度的元素数量是否为奇数，若为奇数则舍弃最后一个元素，确保元素数量为偶数。然后通过 `(quantized[::2] << 4) | quantized[1::2]` 将两个 4 位整数打包到一个 8 位整数中，实现更高的压缩比。

iv. **返回结果**：返回打包后的量化数据、缩放因子和零点。

v. **反量化过程**：

1. **数据解包**：先将打包后的量化数据展平为一维数组，再通过位运算将一个 8 位整数解包为两个 4 位整数，恢复成量化前的数组形状。
2. **反量化操作**：与 int8 反量化类似，利用存储的缩放因子和零点，通过公式 `(unpacked.astype(np.float32) - zero_point) * scale` 将解包后的 4 位整数还原为浮点数。

# 实验结果

1. 数据集：实验数据集来自flickr8k，这是一个图文配对数据集，每张图片对应着一句英文描述。含有8091对图文。随机选取6069对训练数据，2022对测试数据。
2. 模型：多模态嵌入模型选取openai/clip-vit-base-patch32，该模型给全体图片以及文本分配512维向量，供后续三种对比方法进行压缩。多模态理解模型选取Salesforce/blip-vqa-base。
3. 评估指标：测试集上文搜图以及图搜文的平均搜索准确率。
   a. 用clip模型得到所有图片和文本对各自的clip嵌入向量（512维）（参考main.py的main函数）
   b. 首先计算测试集上每张图片按照余弦相似度（内积计算）检索得到的对应最相似（top1）的文本，作为图搜文的结果。对测试集上每张图片都可以得到一个检索的文本，统计测试集图片当中成功检索到正确的文本的比例，作为图搜文正确率。（参考cca.py的eval_final函数）
   c. 其次计算测试集上每个文本按照余弦相似度（内积计算）检索得到的对应最相似（top1）的图片，作为文搜图的结果。对测试集上的每个文本都可以得到一个检索的图片，统计测试集文本当中成功检索到正确图片的比例，作为文搜图正确率。（参考cca.py的eval_final函数）
   d. 最终的评估指标是图搜文正确率和文搜图正确率的平均值。由于有三种压缩方法（PCA,CCA,VQ-4bit,VQ-8bit）和不压缩的baseline方法，则总共有5组测试集结果。
4. 实验设置
   a. PCA：将训练集的文本表征（shape: 6069*512）和图片表征(6069*512)拼接得到形状(12138*512)的矩阵，并用主成分分析提取主成分。通过经过超参数调整，设置累计方差贡献率为0.999，保留维数为492维（492个主成分），得到了降维矩阵形状为(512,492)。将该降维矩阵用于测试集的图片和文本表征的降维，从而实现向量压缩。并在PCA压缩后的向量空间进行平均搜索准确率的计算。（代码：pca.py）

b. CCA：利用训练集的文本表征（shape: 6069*512）和图片表征(shape:6069*512)做训练寻找W1和W2分别作为图片和文本的投影矩阵，在降维的同时使得降维后的文本和图片向量相关性尽可能大。此处设定子空间的维数为256，即减少一半的存储开销。在训练集求解W1和W2后用于测试集上图片和文本的表征压缩实现降维，并在CCA降维后的空间进行平均搜索准确率的计算。（代码：cca.py）

c. VQ-8bit：在训练集和测试集上做相同的步骤：对数据集上的图片和文本表征分别做量化（即文本表征和图片表征有各自独立的scale和zero_point）用于节省存储：虽然维数未变但是精度从32维降为8位节省为原始的1/4。等到需要搜索时，再将量化后的文本和图片嵌入做反量化（时间换空间）并进行搜索，进行平均搜索准确率的计算（代码：vq.py）

d. VQ-4bit：与VQ-8bit相同，但是存储节省为原来的1/8.

5. 延伸：Multimodel-RAG-VQA全流程。

a. Multimodal-Retrieval。以query“a picture of a cat”为例，不做表征压缩，检索到了50030244_02cd4de372.jpg

b. VQA：将图片和问题一起送入blip-vqa-base模型。

i. 图片+“what is the color of the cat?”下，模型回复“gray”

ii. 图片+“How many cats are there in the picture?”下，模型回复"3"

iii. 图片+“Where is the cat on?”下，模型回复"grass"

c. 总结来看，图片检索结果仍旧不精确（图中有多只猫），且vqa回答较短。

6. 结果

| | 训练集文搜图 | 训练集图搜文 | 训练集<br>平均正确率 |
|---|---|---|---|
| 基线 | 0.3098 | 0.3249 | 0.3174 |
| PCA | 0.0873 | 0.1331 | 0.1102 |
| CCA | 0.8522 | 0.8426 | 0.8474 |
| VQ-8bit | 0.2356 | 0.0348 | 0.1352 |
| VQ-4bit | 0.2177 | 0.1417 | 0.1797 |

| | 测试集文搜图 | 测试集图搜文 | 测试集<br>平均正确率 |
|---|---|---|---|
| 基线 | 0.4703 | 0.4688 | **0.4696** |
| PCA | 0.1103 | 0.2062 | 0.1583 |
| CCA | 0.3126 | 0.2918 | 0.3022 |
| VQ-8bit | 0.3783 | 0.0900 | 0.2342 |
| VQ-4bit | 0.3571 | 0.2379 | 0.2975 |

| | 准确率下降 | 节省存储 |
|---|---|---|
| 基线 | 0.00% | 0.00% |
| PCA | 66.30% | 3.91% |
| CCA | **35.65%** | 50.00% |
| VQ-8bit | 50.13% | 75.00% |
| VQ-4bit | 36.65% | **87.50%** |

# 结论

1. 在这个任务下clip模型生成的表征在全量空间下也只能达到46.96%的平均搜索正确率，即这个任务具有一定的难度。大部分情况下图搜文会比文搜图指标更低更难，这或许是因为图片表征不够精确的原因。
2. 三个方法中：
    a. 4位的向量量化方法综合来看最优，缺点是会有额外的测试时耗时。首先是VQ-4bit在存储节省87.50%下，效果仅下降了36.65%。但是由于在搜索时需要将量化后的向量再次反量化，需要额外的推理延时。由于反量化后的结果不会存储，因此该方法适合训练图片多但是测试请求少的场景。

b. CCA效果也不错，但是缺点是有额外的训练耗时。其节省了50%的存储，准确率下降了最少的35.65%，但是其需要在训练集上计算两个投影矩阵，如果训练图片过多则在训练侧会带来额外耗时，或者如果测试图片和训练图片存在分布偏移则该方法效果会大幅下降。因此该方法适合训练图片较少、测试请求较多的场景。

c. PCA效果较差，相比于CCA利用了图片文本对的信息交互，PCA只能利用矩阵本身的特征值信息，去除冗余的主成分从而实现降维（尽管实验中图片和文本矩阵做了拼接一起降维）。这样的方法没有考虑到图片和文本的内积交互的要求，因此降维后搜索效果很差。

3. 延伸：

a. 从本质上，clip等对比学习的思想就是典型相关分析，即clip利用对比损失训练的两个adapter做文本和图片表征的映射从而使得对应的图片文本距离更近、非对应图片文本距离更远，这种思想就是借鉴的统计中典型相关分析的思想。

$$\mathcal{L}_{CLIP} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\text{sim}(z_i,t_i)/\tau)}{\sum_{j=1}^{N}\exp(\text{sim}(z_i,t_j)/\tau)}\right) - \frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\text{sim}(t_i,z_i)/\tau)}{\sum_{j=1}^{N}\exp(\text{sim}(t_i,z_j)/\tau)}\right)$$

b. 区别：典型相关分析通过特征值求解得到投影矩阵的解析解，而对比学习的想法是用对比损失计算梯度进行梯度下降更新参数。此外对比学习还利用了负样本的距离更大（在分母中实现了in-batch negative的信息）这一要求，但是典型相关分析则没有考虑到非相关样本相关系数要尽可能小这一正则约束。