|  |  |
| --- | --- |
| **机器学习实验报告** | |
| 第 4 次 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
| **姓名** | 何铭韬 |
| **班级** | 工设2204班 |
| **学号** | 2223312318 |
| **电话** | 13034642319 |
| **Email** | 2780530464@qq.com |
| **日期** | 2025-06-16 |

目录

[1实验题目 3](#_Toc21035)

[2实验目的 3](#_Toc16034)

[3实验环境与工具 3](#_Toc3269)

[4实验背景与理论 3](#_Toc28084)

[5核心代码 3](#_Toc5910)

[5.1 TODO 1 3](#_Toc27602)

[5.2 TODO 2 3](#_Toc6818)

[5.3 TODO 3 3](#_Toc26649)

[5.4 TODO 4 3](#_Toc3889)

[6实验结果 4](#_Toc21467)

[7总结与收获 4](#_Toc24248)

# 1实验题目

使用 k-means 算法和 PCA 降维算法在 Holiday 数据集上实现基于内容的图像检索

# 2实验目的

（对本次实验任务进行理解，明确实验需掌握的技能）

# 掌握 k-means 聚类算法的实现原理；

# 理解主成分分析（PCA）在降维和数据预处理中的应用；

# 掌握三角化嵌入（triangulation embedding）方法与检索特征的生成；

# 熟悉基于相似度度量的图像检索流程，理解 mAP（mean Average Precision）评估指标。

# 3实验环境与工具

（填写本机或实验室服务器的硬件配置，列出操作系统、Python 版本及核心第三方库）

硬件环境：Intel® Core™ i5-13700kf，NVIDIA GeForce RTX 4070super, 32 GB 内存

操作系统：Windows 11 64 位

开发语言：Python 3.9

核心第三方库：

MindSpore 2.0（张量运算与并行加速）

NumPy 1.24（数值计算）

SciPy 1.10（数据加载）

mat73（Matlab 数据读取）

tqdm（进度条显示）

自定义模块：trie.py、embedding.py、eval\_.py

# 4实验背景与理论

**k-means 聚类：**将大量的 SIFT 描述子（128 维）在 Flickr60k 数据集上聚类，得到 k 个视觉词中心，用于后续的三角化嵌入。

**三角化嵌入（Triangulation Embedding）：**将每张图片的局部特征投影到 k 个聚类中心，得到 k×128 的高维向量，再减去全局均值并求和，以增强判别能力。

**PCA 降维：**对三角化嵌入后的特征进行主成分分析，保留高方差方向，得到投影矩阵P，将高维特征映射到较低维空间，减小计算与存储开销。

**图像检索流程：**

在Flickr60k 上用 k-means 训练聚类中心与 PCA 参数；

对Holidays数据集每张图像提取 SIFT 描述子并进行三角化嵌入、PCA 投影，得到检索特征；

对查询集与图库特征计算相似度，返回 Top-K；

计算mAP评估检索性能。

# 5核心代码

(请勿粘贴截图)

## 5.1 TODO 1

（计算每个点与聚类中心之间的距离，并重新划分簇）

## **def calc\_clusters(data\_points: ms.Tensor, centroids: ms.Tensor, k: int) -> ms.Tensor:**

## **reshape = ops.Reshape()**

## **argmin = ops.Argmin()**

## **# 重塑 centroids 为 (1, k, feat)，再沿批次复制到 (num\_pts, k, feat)**

## **centroid\_matrix = ops.repeat\_elements(**

## **reshape(centroids, (1, k, data\_points.shape[1])),**

## **data\_points.shape[0],**

## **axis=0**

## **)**

## **# 计算平方欧氏距离**

## **dists = ops.ReduceSum()(ops.Square()(data\_points.reshape(-1,1,data\_points.shape[1]) - centroid\_matrix), axis=2)**

## **# 每个点分配到距离最近的中心**

## **centroid\_group = argmin(dists, axis=1)**

## **return centroid\_group**

## 5.2 TODO 2

（计算新的聚类中心）

**def data\_group\_avg(group\_ids: ms.Tensor, data: ms.Tensor, k: int):**

**# group\_ids 形如 (num\_pts,), data 形如 (num\_pts, feat)**

**unsorted\_segment\_sum = ops.UnsortedSegmentSum()**

**# 统计每个簇内点的总和 & 个数**

**sum\_by\_group = unsorted\_segment\_sum(data, group\_ids, k)**

**ones = ops.OnesLike()(group\_ids)**

**count\_by\_group = unsorted\_segment\_sum(ones, group\_ids, k).reshape((k,1))**

**# 平均化得到新中心**

**avg\_by\_group = sum\_by\_group / count\_by\_group**

**return avg\_by\_group**

## 5.3 TODO 3

（求取特征均值 x\_mean）

## **def trie\_learn(data: ms.Tensor, centers: ms.Tensor):**

## **n\_samples, n\_features = data.shape**

## **k = centers.shape[0]**

## **x\_mean = ops.Zeros()((k \* n\_features,), dtype=ms.float32)**

## **for i in range(k):**

## **# 对第 i 类簇的所有归一化差值求和**

## **diff = data - centers[i]**

## **normed = diff / ops.norm(diff, dim=1, keepdim=True)**

## **x\_mean[i\*n\_features:(i+1)\*n\_features] = ops.ReduceSum()(normed, axis=0)**

## **x\_mean = x\_mean / n\_samples**

## **# … 后续协方差、PCA 计算略**

## **return x\_mean, eigvec, eigval**

## 5.4 TODO 4

（补全三角化嵌入）

**def trie\_suma(data, centers, x\_mean):**

**kc, f = centers.shape[0], data.shape[1]**

**# h\_y 形状为 (n\_descs, kc\*f)**

**h\_y = ops.Zeros()((data.shape[0], kc \* f), dtype=ms.float32)**

**for i in range(kc):**

**diff = data - centers[i]**

**norm = ops.norm(diff, dim=1, keepdim=True) + 1e-8**

**h\_y[:, i\*f:(i+1)\*f] = diff / norm**

**# 去均值并累加得到图像级特征**

**h\_y = h\_y - x\_mean**

**return h\_y.sum(axis=0)**

# 6实验结果

**结果(文字)：**

Holidays k = 2 d = 128 pw = 1.00 map = 0.358

Holidays k = 2 d = 128 pw = 0.70 map = 0.383

Holidays k = 2 d = 128 pw = 0.50 map = 0.392

Holidays k = 2 d = 128 pw = 0.30 map = 0.369

Holidays k = 2 d = 128 pw = 0.20 map = 0.350

Holidays k = 2 d = 128 pw = 0.00 map = 0.272

Holidays k = 4 d = 384 pw = 1.00 map = 0.550

Holidays k = 4 d = 384 pw = 0.70 map = 0.564

Holidays k = 4 d = 384 pw = 0.50 map = 0.548

Holidays k = 4 d = 384 pw = 0.30 map = 0.531

Holidays k = 4 d = 384 pw = 0.20 map = 0.518

Holidays k = 4 d = 384 pw = 0.00 map = 0.518

Holidays k = 8 d = 896 pw = 1.00 map = 0.595

Holidays k = 8 d = 896 pw = 0.70 map = 0.603

Holidays k = 8 d = 896 pw = 0.50 map = 0.599

Holidays k = 8 d = 896 pw = 0.30 map = 0.594

Holidays k = 8 d = 896 pw = 0.20 map = 0.578

Holidays k = 8 d = 896 pw = 0.00 map = 0.543

**结果描述和分析：**

1. 聚类数 k 增大，使得MAP（Mean Average Precision，图像检索的平均准确率）提升

k=2 时，最高 MAP ≈ 0.392

k=4 时，最高 MAP ≈ 0.567

k=8 时，最高 MAP ≈ 0.603

分析：

1）增加聚类数 k 意味着视觉词的数量增加，有助于更细致地区分图像的局部特征，从而提高检索准确率；

2）但同时也会带来更高的计算成本和维度（d）。

2. Power-law normalization 对性能有积极作用

观察每组中 pw 从 1.00 到 0.00 的变化：

1）k = 2:MAP 从 0.358上升到 0.392（pw=0.50）后下降到 0.272，最优pw ≈ 0.5；

2）k = 4:MAP稳定在 0.563~0.567 区间，pw=0.7时略优；

3）k =8:MAP随 pw 减小先略增后略降，pw = 0.7 时最优（MAP=0.603）。

分析：

1）Power-law normalization 可以缓解某些特征值偏大对整体相似度的负面影响；

2）适当的 pw（通常在 0.5~0.7 之间）有助于提升特征的区分度；

3）当pw=0 时不做归一化，性能明显下降（例如 k=2 时 MAP 跌至 0.272）

1. 结论：

1）更大的聚类数 k 能显著提升图像检索的准确率；

2）Power-law normalization 是提升性能的关键操作，特别是在 k 较小时对性能影响更大；

3）最优的 pw 值在 0.5~0.7 之间，建议在实际应用中进行调优；

4）无需归一化（pw=0.00）会显著降低系统性能，应避免使用。

# 7总结与收获

（用 3–5 条要点形式总结本次实验的主要收获）

成功实现了 k-means 聚类与三角化嵌入，掌握了簇中心初始化、距离计算与中心更新流程；

运用 PCA 对高维特征进行降维，加速了检索计算且保持了检索精度；

理解了三角化嵌入中“减去全局均值并求和”步骤的作用，有效增强了图像描述子的一致性；

通过 mAP 指标评估检索系统性能，发现 k=4 时能获得最佳平衡；

熟悉了整个基于内容的图像检索流水线，为后续优化（如加速索引、量化压缩）打下基础。