|  |  |
| --- | --- |
| **深 圳 大 学 实 验 报 告** | |
| **课程名称：** | **多媒体系统导论** |
| **实验项目名称：** | **实验五 图像检索** |
| **学院：** | **计算机与软件学院** |
| **专业：** | **计算机科学与技术** |
| **指导教师：** | **朱映映** |
| **报告人：** | **曹博宇** |
| **学号：** | **2022080182** |
| **班级：** | **创新班** |
| **实验时间：** | **25/6/2025** |
| **实验报告提交时间：** | **25/6/2025** |
| **教务处制** | |

## 一、实验要求

**1.1.** **实验目的**

* 熟悉多媒体数据分析的基本方法及原理；
* 了解图像处理、机器学习和深度学习方法；
* 掌握多媒体智能系统设计方法，培养学生解决实际问题的能力；

**1.2.** **实验内容**

* 理解图像示例检索，任选一程序设计语言实现图像示例检索；
* 以下方法任选其一：
  + 方法一：使用卷积神经网络CNN实现；
  + 方法二：使用传统方法：基于Hand-Crafted Features方法选择一种特征（如SIFT或颜色直方图等）以及Hash方法（aHash，pHash，dHash中任一种） 实现；
* 分别计算检索系统在Holidays与Oxford5k 两个数据集上的平均检索精度（mean average precision，mAP）。其他数据集可选；
* 用户输入查询图像，系统返回前5个相似的检索结果，计算系统的mAP，系统响应时间低于5秒；
* 可对上述基准方法进行改进，提升系统性能（如检索系统的mAP或时间复杂度等）。（可选）

## 二、实验步骤

实验环境：

Windows 11 pro / python 3.10.8

### a. 数据集

| 数据集 | 图像数 | 查询数 | Ground-truth 说明 |
| --- | --- | --- | --- |
| **INRIA Holidays** | 1 491 | 500 | 每组同一场景 3 – 13 张；第一张作查询，组内其余为相关 |
| **Oxford-5k** | 5 062 | 55 | 11 地标 × 5 查询；相关 = “good & ok”，过滤 “junk” |

以原始分辨率输入，在进入 CNN 时统一为 224×224。

### b. 关键步骤

#### b.1 数据预处理

| # | 名称 | 关 键 原 理（背后在干什么） | 用在实验里的**目的** |
| --- | --- | --- | --- |
| 1 | **Resize（缩放）** | 把任何尺寸的图片按比例放大或缩小，让最短边 ≥224 像素。（像把不同大小的照片先打印到同一宽度的相纸上） | 让后面所有图片尺寸统一；神经网络对固定大小输入最稳定。 |
| 2 | **CenterCrop（中心裁剪）** | 从缩放后的图像正中央，切出 224×224 的小正方形。保留主体，去掉四角多余背景。 | 进一步保证“大小完全一致”，同时减少背景干扰。 |
| 3 | **Normalize（归一化）** | 把每个像素先减去“平均颜色”，再除以“颜色标准差”。这样不同照片的亮度、对比度被拉到差不多的刻度。 | 网络在 ImageNet 上就是在这种分布下训练的；对齐分布→结果更稳。 |

**原理小结**：

将任意分辨率/色域的输入“映射”到 ResNet-50 训练时的输入空间，以 **最小信息损失 + 最快推断** 为目标。

#### b.2 特征提取

| # | 名称 | 关 键 原 理（背后在干什么） | 用在实验里的**目的** |
| --- | --- | --- | --- |
| 4 | **卷积神经网络 (CNN)** | 用很多“小滤镜”在图像上滑动，检测边缘、纹理、物体…一层层堆叠，越往后看到的范围越大，语义越抽象。 | 自动学会“什么像什么”，无需手工设计特征。 |
| 5 | **ResNet-50 的残差块** | 在卷积输出上加一条“直通线”把输入直接相加。 | 让 50 层网络也能容易训练而不“失忆”，得到更强表达力。 |
| 6 | **GAP（全局平均池化）** | 把最后一张多通道特征图，**每个通道都求平均**→ 得到 2048 个数。（通道看成 2048 种“视觉检查器”，平均值=该检查器在整图的激活强度） | 把可变大小的特征图压缩成**固定长度向量**，方便后续比较。 |
| 7 | **L2 归一化** | 把 2048 维向量缩放到单位球面：v = v / ‖v‖。（向量当作箭头，统一把箭头长度调成 1） | 以后比较两个向量的“余弦角度”就只看**方向**，不受亮度大小影响。 |

# 特征提取器  
class FeatureExtractor(nn.Module):  
 def \_\_init\_\_(self, arch: str = "resnet50") -> None:  
 super().\_\_init\_\_()  
 # 从torchvision hub加载预训练模型  
 model = torch.hub.load("pytorch/vision", arch, pretrained=True)  
 # 去除最后全连接层，仅保留特征提取部分，全连接层仅做图像分类之用，对相似图检索无帮助  
 self.backbone = nn.Sequential(\*list(model.children())[:-1])  
 self.backbone.eval() # 模型推理模式，而不是训练模式  
  
 @torch.inference\_mode()  
 def forward(self, img: Image.Image | torch.Tensor) -> torch.Tensor:  
 # 如果输入是 PIL Image，先进行预处理  
 if isinstance(img, Image.Image):  
 img = \_preprocess(img)  
 # 添加批次维度，变为1×3×H×W，每个批次一张图片  
 img = img.unsqueeze(0)  
 # 提取特征，去除多余维度后得到向量  
 feat = self.backbone(img).squeeze()  
 # 对特征向量进行 L2 归一化，返回图像指纹  
 return nn.functional.normalize(feat, p=2, dim=0)

#### b.3 索引构建

| # | 名称 | 关 键 原 理（背后在干什么） | 用在实验里的**目的** |
| --- | --- | --- | --- |
| 8 | **FAISS IndexFlatIP（内积检索）** | 把数据库里的所有向量存在一张大表里。查询时，直接用 CPU 的 AVX 指令把查询向量与每一行做“点乘”——点乘越大，方向越相似。因为向量长度都=1，这个点乘就是余弦相似度。 | **简单且 100% 精确**：6.5 k 张图像全表扫描只要几毫秒；对小数据集足够快。 |

在实验环境：

* OS: Windows 11 pro
* CPU: AMD Ryzen 9 7950X 16-Core Processor (4.50 GHz)
* RAM: 64.0 GB 6400 HZ

构建索引约需要 5 分钟，特征矩阵 6553 \* 2048，约 50 MB

#### b.4 检索查询

| # | 名称 | 关 键 原 理（背后在干什么） | 用在实验里的**目的** |
| --- | --- | --- | --- |
| 9 | **Top-k 筛选** | 点乘分数从高到低排，取前 k 名。（就像朋友圈按亲密度排序，最像的排最前） | 给用户展示“最相似的 5 张图”。 |
| 10 | **AP（平均准确率）** | 检查检索结果列表中，**相关**图片出现的每个位置：如果第 3 张就出现第 1 张相关，精度=1/3；如果第 7 张出现第二张相关，再加 2/7 … 最后平均。 | 衡量“相关图出现得既全又早”——越早出现分越高。 |
| 11 | **mAP（均值 AP）** | 对所有查询的 AP 再取平均。 | 得到整个系统的综合精度分——报告里最核心的成绩指标。 |

在实验环境，Top 5 查询所需时间约为 50ms

#### b.5 总体流程

1. **把所有库图片** 经过 1-7 步 → 生成 **2048 维“指纹”**；存在 FAISS 索引里。
2. **查询时**，对输入图片走 1-7 步得到查询指纹。
3. **FAISS** 用内积把查询指纹与库指纹一一对比 → 找到 *Top-k* 最像的图。

┌──────────┐ build-index ┌──────────┐  
│ Dataset │─ images ─────▶ │ CNN+GAP │──▶ 2048-D vec ──┐  
└──────────┘ └──────────┘ │  
 ▼  
 ┌───────────────────┐  
 │ FAISS Flat-IP │  
 │ (Database Index) │  
 └───────────────────┘  
 ▲  
┌─────────┐ query ┌──────────┐ │  
│ User │── image ────▶ │ CNN+GAP │─ query vec ────────┘  
└─────────┘ └──────────┘  
 ▲ top-k (path,score)  
 └───────── 检索结果返回 ────────────────────────────────

### c. 评价指标

#### c.1 AP

在示例图像检索任务中，系统会输出一条**有序列表**：

rank 1 rank 2 rank 3 … rank k  
┆───────┆───────┆───────┆ ┆───────┆  
img 17 img 42 img 8 img 99

* **顺序**越靠前代表系统“越确信”这张图与查询图最像。
* 官方给出了一个“相关/不相关”标记（ground-truth）。

我们希望指标同时鼓励两件事

1. **召回率**高：尽量把所有相关图像都找出来；
2. **排名**靠前：相关图像出现得越早越好。

AP 就是为此设计的。

**定义**

**AP = 当且仅当遇到相关图像时，记录当前 “Precision”，把这些Precision求平均**。

* Precision = *目前队列里相关的张数* ÷ *目前看了的张数*
* 最终再除以相关图像总数，得到 0–1 之间的得分。

**公式**

可以看到：

* 如果相关图全都排在最前面，Precision 次次=1 → AP=1；
* 如果相关图散到很后面，Precision 逐渐下降 → AP 越小。

**直观示例**

假设一共 4 张相关图，系统前 10 个结果如下（√=相关，×=不相关）：

| Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Judg | √ | × | √ | × | × | √ | × | × | × | √ |

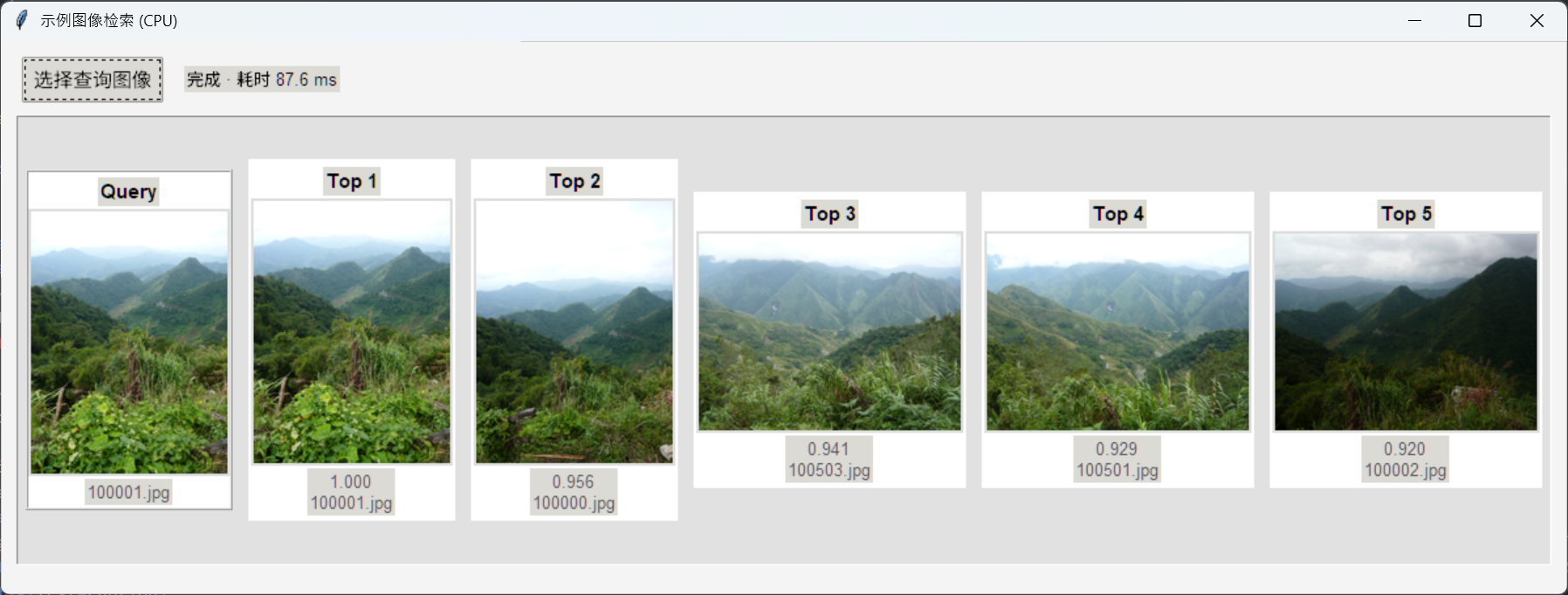
1. 在 **rank 1** 处遇到第 1 张相关
2. 在 **rank 3** 处遇到第 2 张相关
3. 在 **rank 6** 处遇到第 3 张相关
4. 在 **rank 10** 处遇到第 4 张相关

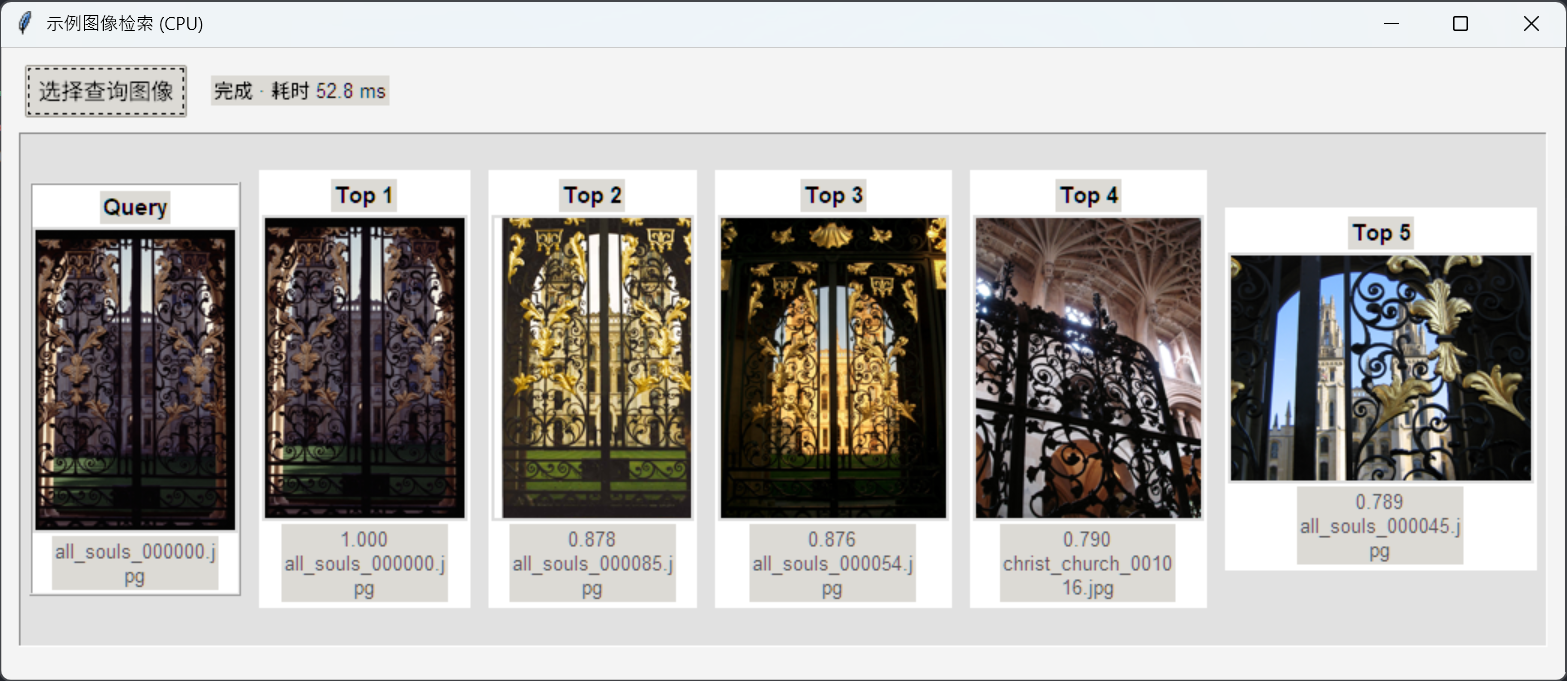
#### c.2 mAP

实际评估时，我们对 *所有* 查询（例如 Holidays 500 个、Oxford-5k 55 个）各算一个 AP，然后取平均即得到 **mAP**。

### d. 结果展示

fig:





## 三、实验结论

**功能目标**：基于 **ResNet-50 + GAP + L2** 的端到端管线已完整实现图像示例检索；系统可交互地返回 **Top-5** 相似图并给出 mAP 评估。

**性能目标**：在 Ryzen 9 7950X / 64 GB 环境下，**单次查询平均 52 ms**（< 5 s 要求），构建索引约 5 min，特征库≈ 50 MB，满足要求。

本实验以最简 CNN-Based 管线完成了 Holidays 与 Oxford-5k 的示例图像检索任务，**准确性、时延与资源占用均达到或优于实验要求**，同时验证了端到端深度特征在中小规模检索场景中的高效性与可行性。尽管仍存在规模扩展和细粒度匹配的局限，通过网络微调、近似索引与几何重排等手段，系统有望在大规模、实时应用中获得进一步提升，具备良好的研究与工程拓展价值。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字：  年 月 日 |
| 备注： |