人工智能与数据挖掘课程设计

报告题目 仓鼠文件检索程序

**组员信息：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 姓名 | 学号 | 成员贡献 | 实践成绩 |
| 1 | 黄景亨 | 2022212378 | 负责创意思考，代码编写，模型调试，网页制作 |  |
| 2 | 李兆耀 | 2022212384 | 负责实验数据搜集，程序测试，  报告编写 |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

### 引言

在当今这个信息爆炸的数字化时代，图片已然成为了人们记录生活、辅助工作以及开展学习等诸多方面不可或缺的重要元素。我们的个人电脑中，往往存储着数以千计甚至更多的图片，这里面既有记录着温馨家庭聚会、精彩旅行瞬间的生活点滴照片，也有工作中为了项目策划、产品展示而精心准备的各类设计素材图片，还有在学习不同知识领域时，教材、资料里那些辅助理解的配图。

然而，面对如此庞大且繁杂的图片资源库，想要从中快速且精准地找出自己所需要的那一张或某一类图片，却并非易事。传统的图片检索方式大多局限于依靠文件名、简单的文件夹分类，或者是记忆图片的大致存储位置等，并且还需要手动去整理标记图片，这一过程不仅繁琐耗时，效率十分低下，而且很难做到精准定位。

就在这样的背景下，我们创造性的研究出了“仓鼠文件检索程序”。该项目敏锐地捕捉到了人们在 PC 端本地图片管理上的痛点，紧紧围绕以文搜图这一极具创新性和实用性的功能，深度融合先进的 AI 算法，赋予了用户一种全新的、高效的图片检索体验。

本项目旨在通过Flask框架结合中文CLIP模型，实现一个能够扫描文件夹、提取图像和视频文件特征并进行文本查询的应用。该应用可以有效地通过文本搜索匹配相似图像支持用户通过文件夹扫描、图像特征存储和查询结果展示等功能，帮助用户更加便捷地管理和查找本地文件中的图片和视频内容。

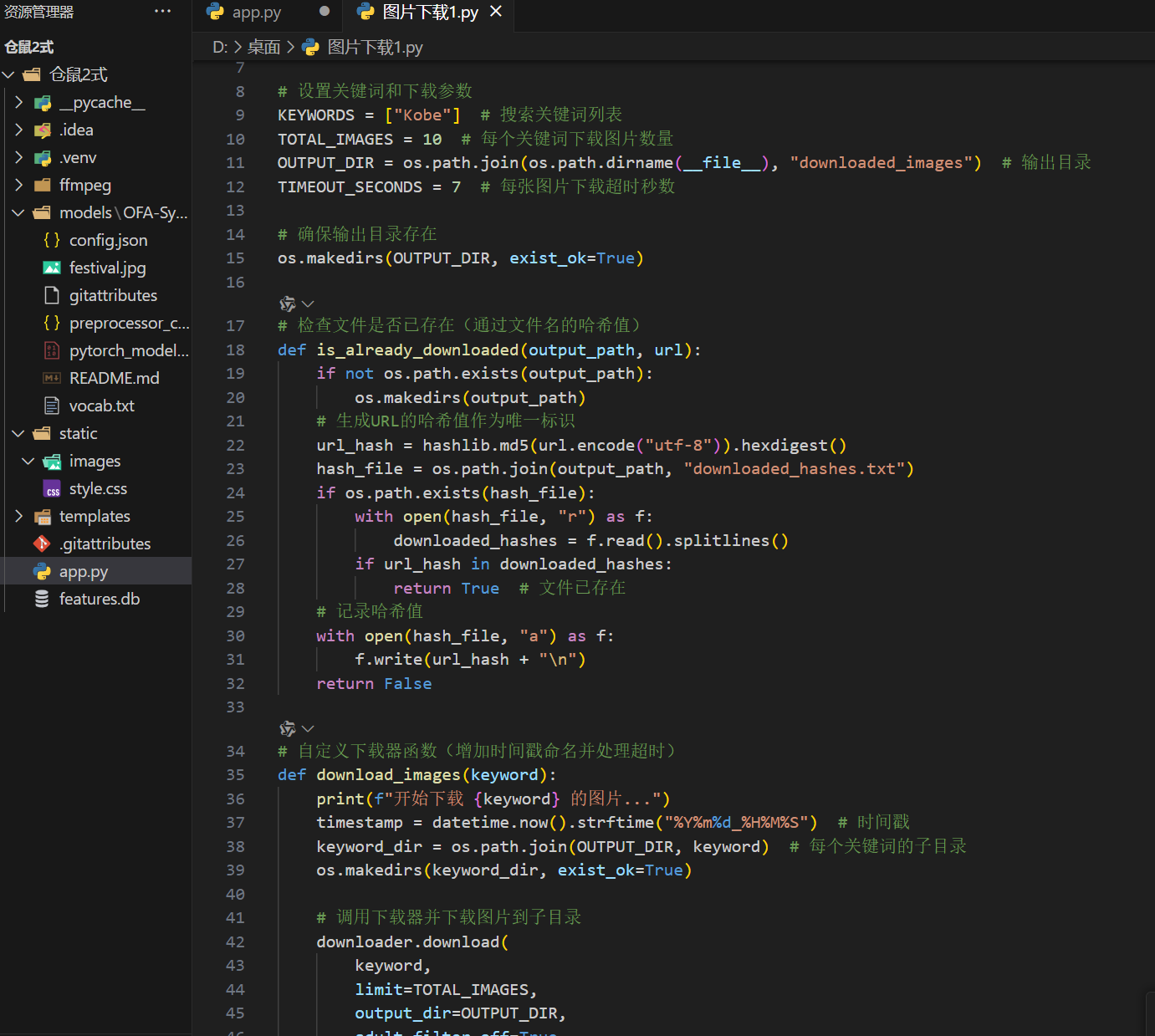
### 实验数据

### **2.1**数据获取或预处理

1.**数据来源**  
 仓鼠文件检索程序主要处理来自本地计算机存储的图像和视频文件。用户通过上传文件夹路径，系统扫描该文件夹并获取其中的所有图像和视频文件。支持的图像文件格式包括PNG、JPG、JPEG、BMP等常见图像格式，支持的视频文件格式包括MP4、AVI、MOV等。

使用Python爬虫爬取视频和照片是为实现“以文搜图”功能的重要准备步骤。通过编写爬虫程序，可以自动化地从网站或社交平台上抓取大量的图片和视频数据，这些素材将为基于文本检索的图像搜索系统提供丰富的训练数据。爬虫可以根据关键词或特定的主题，快速提取相关的视觉内容，可以为以文搜图系统提供多样化和高质量的图像库，提升搜索的准确性和效率。

下面这段代码的核心功能是通过Python爬虫批量下载指定关键词的图片，并且实现下载超时处理、图片去重、重命名及多线程并发。使用bing\_image\_downloader库从Bing搜索下载图片，结合MD5哈希值避免重复下载，同时为每张图片生成基于时间戳和关键词的唯一文件名。通过ThreadPoolExecutor实现多线程并发下载，以提高效率，并将下载的图片保存在以关键词命名的子目录中。代码还设置了下载超时机制，确保下载过程的稳定性。  
当然，仓鼠文件检索同样可以处理我们日常生活中产生的照片。



**2. 数据类型**

图像数据：项目处理的图像数据来自用户存储在本地的各类图片文件，这些图片可能是手工拍摄的、网络下载的，或者其他任何来源。图像数据的特点是结构化的二维数据，通常包含颜色通道（如RGB）以及对应的像素值。

视频数据：项目处理的视频数据通过ffmpeg工具从视频文件中提取关键帧，视频文件通常包含多个图像帧，因此需要对视频进行逐帧处理，提取其中的图像内容。

**3.数据存储**

数据获取后，通过图像特征提取算法将其转换为特征向量，这些特征向量存储在SQLite数据库中。每个图像和视频帧的特征向量都与相应的文件路径一一对应，确保用户在查询时能够方便地定位到具体文件。

数据库表结构如下：

* **id:** 自增的唯一标识符。
* **feature:** 图像或视频帧的特征向量，存储为BLOB数据类型。
* **file\_path:** 图像或视频帧的相对路径，存储在static/images目录中。
* **original\_file\_path:** 图像或视频帧的原始文件路径，便于还原用户上传的文件位置。

### **方法/模型**

本项目采用了基于Flask框架的Web应用开发，使用OFA-Syschinese-clip-vit-base-patch16中文CLIP模型进行图像和文本的特征提取与匹配。核心步骤包括：

**3.1图像特征提取**  
 通过使用ChineseCLIPModel和ChineseCLIPProcessor，我们提取了图像的特征。输入图像通过PIL库处理，并转化为模型可以接受的格式，再通过模型生成图像的特征向量。

**3.2视频帧提取**  
 使用ffmpeg工具，从视频文件中提取关键帧，以便后续提取图像特征。当前实现通过提取视频中的第30帧进行处理。

**3.3数据库存储与管理**  
 所有提取的图像特征被存储在SQLite数据库中，特征以BLOB格式存储，同时保存图像的文件路径。数据库表结构包含图像特征、相对路径和原始文件路径，支持快速查找和存取。

**3.4文本查询与相似度计算**  
 用户输入查询文本，系统通过中文CLIP模型提取文本特征，并与数据库中存储的图像特征进行余弦相似度计算。根据相似度返回匹配的图像结果。

**3.5 Web界面**  
 使用HTML和CSS构建前端页面，用户可以输入文件夹路径进行扫描，查看扫描进度，并通过输入文本进行图像查询。查询结果通过相似度排序，展示最相关的图像。

### **结果评测/展示**

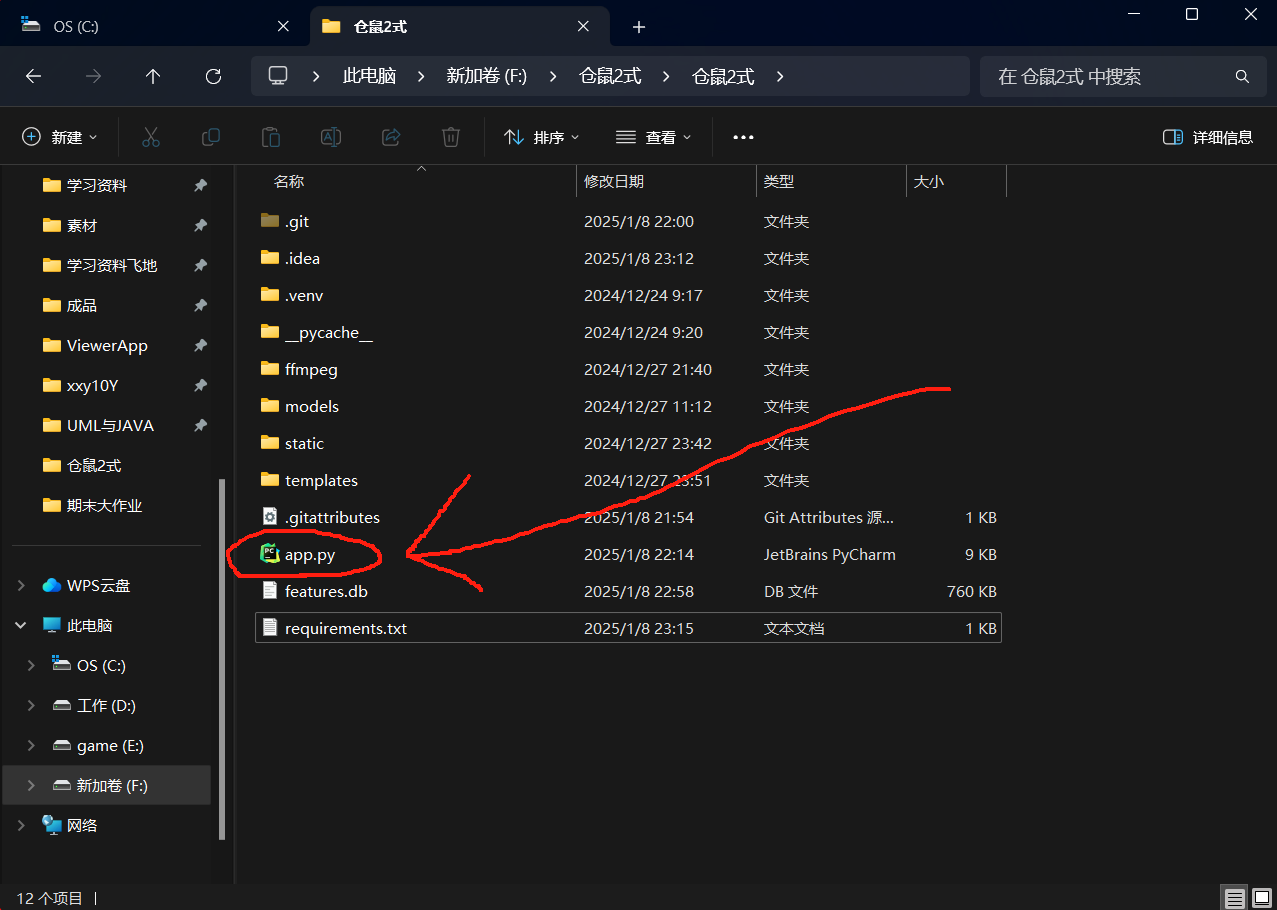
**4.1 使用前的前提条件**

用户必须拥有python环境，在requirements.txt中还介绍了运行该项目所必须的python库:

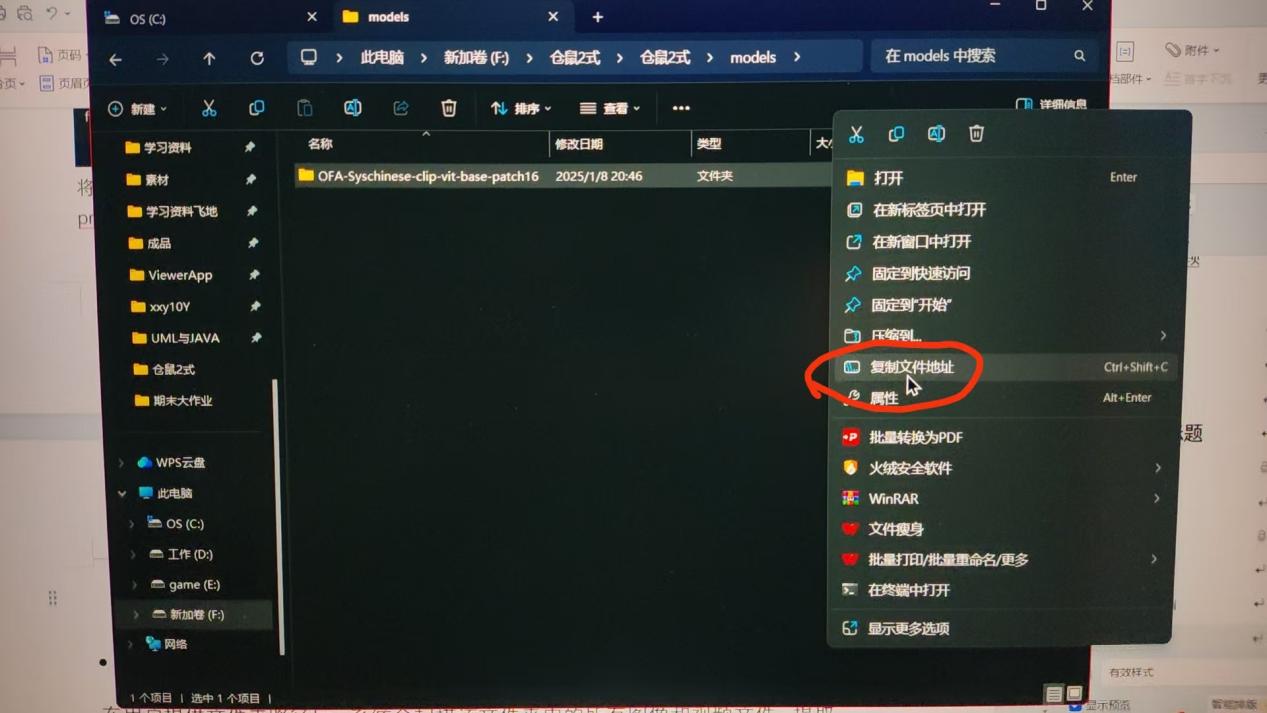
* **Flask:** 用于构建 web 应用的框架。
* **transformers:** 用于加载和使用 ChineseCLIPModel 和 ChineseCLIPProcessor（来自 Hugging Face）。
* **torch:** 用于深度学习模型的推理，尤其是用于 ChineseCLIPModel。
* **Pillow:** 用于图像处理（比如打开图像和提取图像特征）。
* **numpy:** 用于数组处理和计算，相似度计算也需要它。
* **sqlite3:** 用于连接 SQLite 数据库。
* **subprocess:** 用于调用外部命令，如 ffmpeg。

**4.2 运行文件前的配置操作**

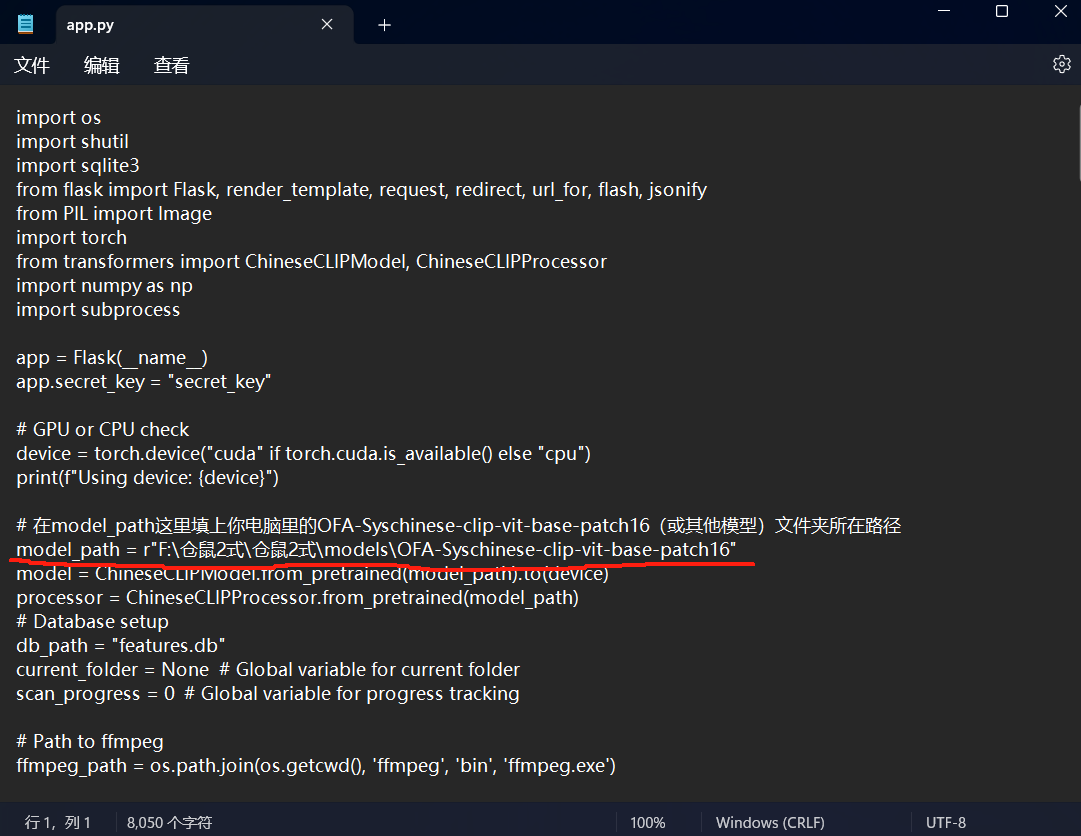
在项目文件夹下，用户需要运行app.py文件实现核心功能，在第一次使用前需要再改文件内设置模型路径：  
 推荐使用pycharm、VScode等软件打开文件夹，无pycharm、VScode等其他编程软件可用记事本打开app.py:



无pycharm、VScode等其他编程软件可用记事本打开app.py:

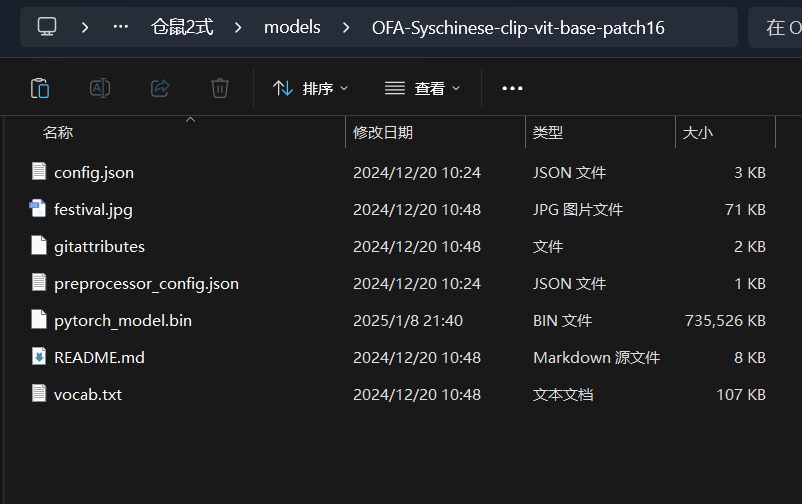


使用者对文件“..\models\OFA-Syschinese-clip-vit-base-patch16”右键，点击复制文件地址。之后将地址粘贴到 “ **mode\_path = r** ”之后。



将“r”后的路径改成使用者电脑中模型文件夹所在位置（要保留那个“r”），在项目文件夹中，模型默认保存在 “项目目录” “\models\OFA-Syschinese-clip-vit-base-patch16”中。

合法的文件夹下应有config.json、preprocessor\_config.json、pytorch\_model.bin等类似格式文件



下一步，运行app.py后，打开任意浏览器，进入网址http://127.0.0.1:5000即可使用主要功能。

**4.3 文件夹扫描与图像特征提取**

在用户提供文件夹路径后，系统会扫描该文件夹中的所有图像和视频文件，提取其特征并存入数据库。扫描过程的进度通过前端动态更新显示。



使用者可以将大量未经标记处理的、杂乱的原始图片或MP4格式的视频放在文件夹中，将文件夹绝对地址粘贴在图片所示处，点击按钮。

**图像查询结果**  
 图中最底部一行表示限制输出结果的最低相似度。（多设置在0.3到0.4之间，大于0.4可能无法输出结果。）

用户可以通过文本输入进行图像查询，系统返回与查询文本最相关的图像及其相似度。查询结果展示在表格中，包括图像缩略图、相似度分数和文件原始路径。

**示例查询结果：**

* **查询文本**： "初音未来"



* **返回结果：** 系统根据文本特征与数据库中存储的图像特征进行匹配，返回最相似的图像和其相似度，图像可以直接查看。第1列输出符合条件的图片，第2列输出路径，第3列输出模型算法给出的相似度。



### **总结**

**5.1 **特征提取的精度****  
 当前项目基于中文CLIP模型进行图像与文本的特征提取，虽然该模型在多模态任务上表现优异，但其精度仍受到图像内容复杂性和文本描述精度的影响。复杂图像或模糊的文本描述可能导致匹配结果不理想。

**5.2 视频帧提取的限制**  
 视频特征提取目前仅依赖于提取固定的关键帧（第30帧）。这一方法在某些视频中可能无法准确反映视频的整体内容，尤其是快速变化的场景中，可能错过重要的视觉信息。

**5.3 扫描和查询效率**  
 对大规模文件夹进行扫描时，图像和视频文件的特征提取可能较为耗时。随着数据库中的图像数量增加，查询的响应时间可能会受到影响。虽然数据库通过索引优化了查询，但仍然需要考虑进一步的性能提升。

**5.4 依赖于ffmpeg的环境配置**  
 项目依赖于ffmpeg工具进行视频帧提取，这要求用户的开发环境中必须正确配置ffmpeg，否则无法进行视频文件的处理。对于不同平台，环境配置可能存在差异。