Content-based Image Retrieval

Kha Le Nhat

Honor Program, Faculty of Computer Science, University of Information and Technology,
Vietnam National University, Ho Chi Minh City

June 4, 2023

Abstract

The aim of every information retrieval (IR) system is to deliver relevant documents to an user's information need (IN). User's need is important to the quality of the system's search results. However, many IR systems ask the users to appreciate their information needs and interact them to the system, usually in form of queries. The systems assume the queries to be a perfect assessment of the information needs and relevant information, ending the interaction. However, in some cases database is very big, which challenges us to find out fast retrieval way. .

Keywords: *Image Retrieval, VGG16, Resnet50, RGBHistogram, LBP, Faiss, Paris6k*

1. Introduction

Content-based image retrieval (CBIR) is a technique used to search for images in a large database based on their visual content, rather than on metadata such as tags or captions. The idea behind CBIR is to extract a set of features from an image, such as color, texture, shape, or other visual characteristics, and use these features to find other images in a database that are visually similar.

The process of content-based image retrieval typically involves several steps. First, the image is preprocessed to remove noise and enhance its visual features. Next, a feature extraction algorithm is applied to the image to extract its visual characteristics. These features can be computed using a variety of techniques, such as color histograms, edge detection, or deep neural networks.

Once the features have been extracted, a similarity measure is used to compare the features of the query image with those of other images in the database. The similarity measure is often based on a distance metric, such as Euclidean distance or cosine similarity, and the aim is to find the images in the database that are most similar to the query image.

The result of a CBIR system is typically a ranked list of images that are most similar to the query image, with the most similar image at the top of the list. CBIR has many applications, such as in image search engines, medical image analysis, and surveillance systems.

Problem modeling:

- **Input:** A collection of information resources (image database), a query image
- **Output:** A ranked list of images that are most similar to the query image, with the most similar image at the top of the list

2. Dataset

The Paris Dataset is a well-known **benchmark dataset** for evaluating content-based image retrieval (CBIR) systems. It contains a collection of 6,412 images of Paris landmarks, such as the La Defense Paris, Eiffel Tower Paris, Hotel des Invalides Paris, Louvre Paris and Moulin Rouge Paris. The images were collected from Flickr and have been manually annotated with ground truth information, such as the name of the landmark, its geographic coordinates, and a textual description.

The Paris Dataset is commonly used in research to evaluate the performance of CBIR systems. The goal of CBIR systems is to retrieve images that are visually similar to a query image, and the performance of these systems can be evaluated using various measures, such as precision, recall, and mean average precision (mAP). The Paris Dataset provides a standardized testbed for evaluating CBIR systems and comparing their performance with each other.

In addition to the original Paris Dataset, several variants and extensions of the dataset have been created, such as the Paris 6k dataset and the Oxford-Paris dataset. These datasets contain variations of the original images, such as cropped or resized versions, and provide additional challenges for evaluating CBIR systems. The Paris Dataset and its variants have become a standard benchmark for CBIR research, and have contributed to the development of more accurate and efficient CBIR systems.

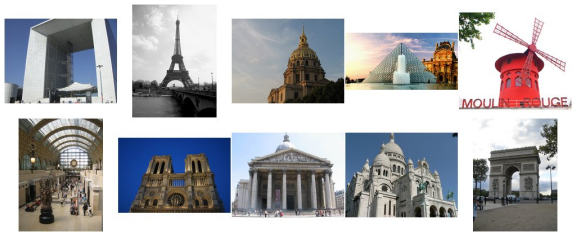

Figure 1. The Paris Dataset

The objective of my research is to improve better methods for searching for specific objects in extremely large datasets of images. In this work I focus on searching large collections of downloaded Flickr images - for Paris dataset.

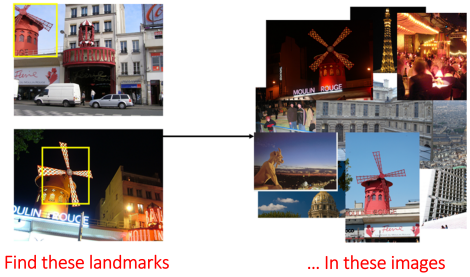

Figure 2. How to query and show result

3. Methods

Below, Figure 3 illustrates the architecture that demonstrates how to query to information retrieval system. Both offline and online are mentioned. For more details, I will explain feature extraction and indexing because they are two main steps which affect the performance of system. Especially, I use the state of the art model and faiss technique from Facebook to improve query system.

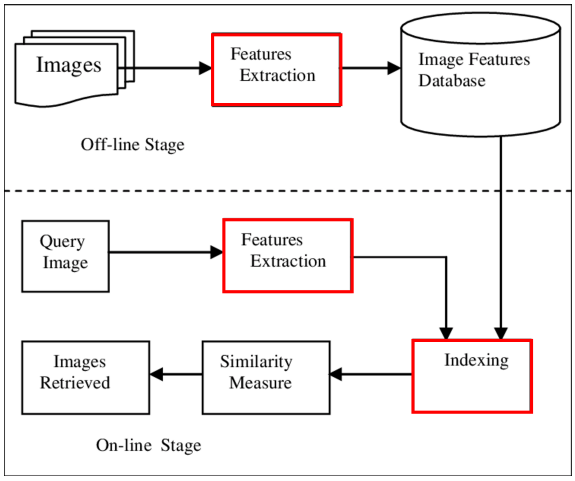

Figure 3. Offline and Online form - Query architecture

3.1. Feature Extraction

Feature extraction is a critical step in content-based image retrieval (CBIR) systems, where the goal is to extract a set of visual features from an image that can be used to represent the image's content. The choice of features to extract can depend on the type of image and the application of the CBIR system. In general, visual features can be divided into three types: low-level features, mid-level features, and high-level features.

I will conduct testing on 4 methods: 2 low-level features (RGBHistogram, Local Binary Pattern - LBP) and 2 high-level features (VGG16 network, Resnet50 network). The following explaination explicit the effectiveness of them.

3.1.1. RGBHistogram

Here are some advantages of using RGBHistogram for feature extraction in the CBIR system:

Color Information: RGBHistogram captures the color information of an image by computing the color distribution of the image in the red, green, and blue color channels. This makes it a powerful feature for image retrieval tasks that rely on color information, such as finding images that are visually similar in terms of their color palette.

Low-dimensional Representation: RGBHistogram produces a low-dimensional feature vector that can be easily compared with other feature vectors using a distance metric, such as Euclidean distance or cosine similarity. This makes it efficient to store and retrieve large numbers of images in a CBIR system.

Robustness to Noise: RGBHistogram is robust to noise and small variations in image content, as long as the overall color distribution of the image remains similar. This makes it a good feature for image retrieval tasks where small variations in image content are expected, such as searching for images that are similar in color, but may have different objects or scenes.

Easy to Implement: The computation of RGBHistogram is simple and fast, and does not require any deep learning frameworks or pre-trained models. It can be easily implemented using standard image processing libraries and functions, making it accessible to researchers and practitioners with basic programming skills.

Applicability to a Wide Range of Images: RGBHistogram can be applied to a wide range of images, including natural images, medical images, and satellite images. This makes it a versatile feature that can be used in a variety of image retrieval applications.

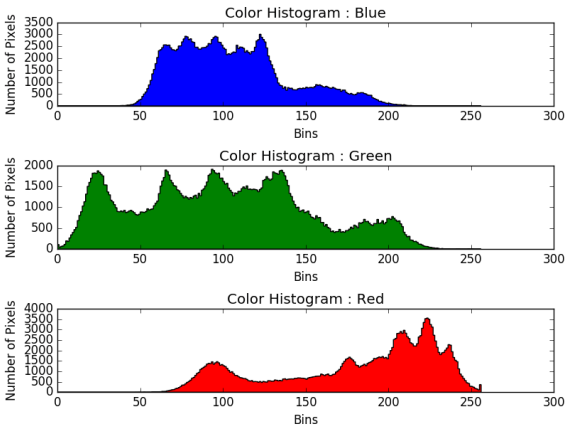

Figure 4. RGB Histogram

3.1.2. Local Binary Pattern

Here are some advantages of using Local Binary Pattern (LBP) for feature extraction in the CBIR system:

Local Texture Information: LBP captures the local texture information of an image by computing the local binary pattern of each pixel. This makes it a powerful feature for image retrieval tasks that rely on texture information, such as finding images that have similar textures, such as grass, water, or fabric.

Invariance to Rotation and Scale: LBP is invariant to rotation and scale, which makes it a good feature for image retrieval tasks that require matching similar textures, regardless of their orientation or size.

Low-dimensional Representation: LBP produces a low-dimensional feature vector that can be easily compared with other feature vectors using a distance metric, such as Euclidean distance or cosine similarity. This makes it efficient to store and retrieve large numbers of images in a CBIR system.

Easy to Implement: The computation of LBP is simple and fast, and does not require any deep learning frameworks or pre-trained models. It can be easily implemented using standard image processing libraries and functions, making it accessible to researchers and practitioners with basic programming skills.

Applicability to a Wide Range of Images: LBP can be applied to a wide range of images, including natural images, medical images, and satellite images. This makes it a versatile feature that can be used in a variety of image retrieval applications.

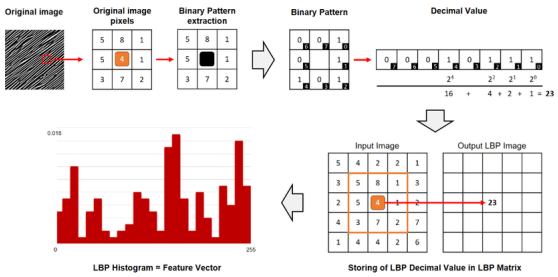

Figure 5. Local Binary Pattern

3.1.3. VGG16

There are several advantages to using VGG16 for feature extraction in content-based image retrieval systems:

High Accuracy: VGG16 has been shown to achieve state-of-the-art performance in image classification tasks, and can also provide high accuracy in feature extraction for content-based image retrieval.

Transfer Learning: VGG16 can be used as a pre-trained model for transfer learning. The pre-trained weights of the VGG16 network can be used to initialize the weights of a new neural network, which can then be fine-tuned for a specific task, such as image retrieval. This can significantly reduce the amount of training data required and can speed up the training process.

Robustness to Variations: VGG16 has been trained on a large dataset of images with a wide range of variations in lighting conditions, object poses, and backgrounds. As a result, VGG16 can provide robust feature representations that are invariant to these variations.

High-level Features: The higher layers of VGG16 capture more abstract and complex features, such as object parts or scene categories. These high-level features can be used to distinguish between different image categories and can help to improve the performance of content-based image retrieval systems.

Availability: The VGG16 network is widely available and can be easily accessed through deep learning frameworks such as TensorFlow and Keras. This makes it easy for researchers and developers to use VGG16 for feature extraction in their own content-based image retrieval systems.

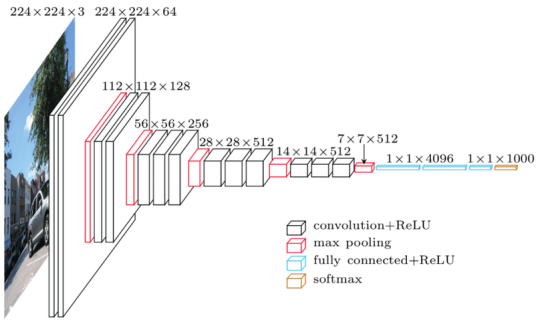

Figure 6. VGG16 architecture

3.1.4. ResNet50

ResNet50 is a deep convolutional neural network that can also be used for feature extraction in content-based image retrieval systems. Here are some advantages of using ResNet50 for feature extraction:

Depth and Complexity: ResNet50 is a very deep network with 50 layers, which allows it to capture complex visual patterns and high-level image features. It has been shown to achieve state-of-the-art performance on many computer vision tasks, including image classification, object detection, and segmentation.

Residual Connections: ResNet50 uses residual connections, which allow the network to learn a residual mapping between the input and output of each layer. This helps to alleviate the vanishing gradient problem and allows for the network to be trained more effectively.

Transfer Learning: Like VGG16, ResNet50 can also be used for transfer learning. The pre-trained weights of the ResNet50 network can be used to initialize the weights of a new neural network, which can then be fine-tuned for a specific task, such as image retrieval.

Robustness to Variations: ResNet50 has been trained on a large dataset of images with a wide

range of variations in lighting conditions, object poses, and backgrounds. As a result, ResNet50 can provide robust feature representations that are invariant to these variations.

Availability: The ResNet50 network is also widely available and can be easily accessed through deep learning frameworks such as TensorFlow and Keras.

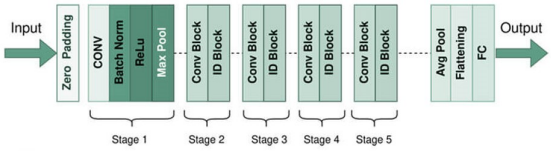

Figure 7. ResNet50 architecture

3.2. Indexing and similarity search

3.2.1. Introduction to Faiss

My goal is a set of relevant image, which is indirectly implemented throughout calculating similarity measures. Similarity measure is based on indexing technique. Indexing is the way to get an unordered set of image into an order that will maximize the query's efficiency while searching or retrieval.

Faiss is a library for efficient similarity search and clustering of dense vectors. It consists of searching algorithms and implements in sets of vectors of any size, up to ones that possibly do not fit in RAM. It supports source code for evaluation and parameter tuning. Faiss includes the most useful algorithms are implemented on the GPU.

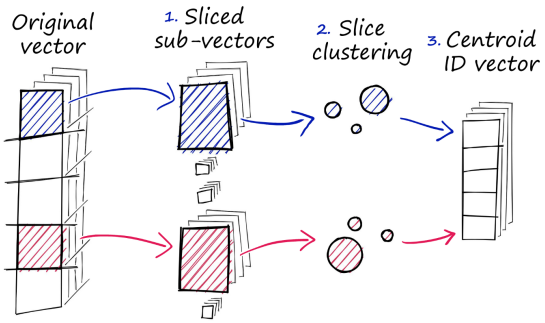

Figure 8. Facebook AI Search Similarity - Faiss

Faiss also can return not just the nearest neighbor, but also the 2nd nearest, 3rd, ..., k-th nearest neighbor. It searches several vectors at a time rather than one (multi-index or batch processing). For many index types, this is faster than searching sequentially.

There is a trade-off between speed and memory. If I accept wrong result 10 percentage of the time, I will have a method that is 10x faster or save 10x less memory. Faiss stores index on disk instead of RAM, especially index is formed binary vectors compared to floating-point vectors.

3.2.2. Main foundations used in Faiss

Inverted index: It is a data structure that searches similar object quickly. Faiss uses inverted index to store feature vectors and their position in feature space. This helps us to search nearest vectors fast.

Quantization: It is a generic method that relates to the compression of data into a smaller space. Quantization is used to reduce the memory footprint of indexes. This is an important

task when comparing large arrays of feature vectors as they must all be loaded in memory to be compared. From that, I can improve the acceleration of retrieval.

Product Quantization: This part applies reducing dimensionality in feature space. Faiss use product quantization with the aim that feature vectors is divided smaller components or subvectors. After that, k-means clustering is used to find the centroids with components, respectively.These centroids with unique IDs and each ID represents a centroid. This helps us to reduce the size of dataset and speed up for searching.

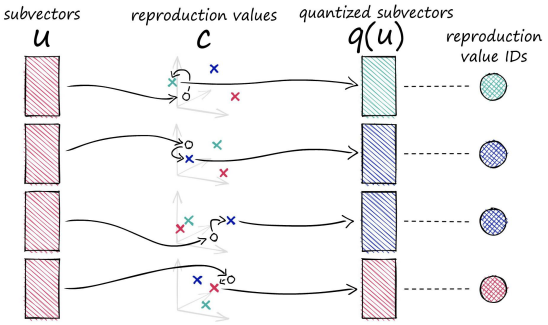

Figure 9. Quantization technique

4. Experiment

I test the system on the Paris dataset, including the test set (query and groundtruth). From the image query, the system will list out a list of 11 images that are most similar to the image query. Then, based on the provided groundtruth, I use the mAP measure to evaluate the efficiency of the system. To facilitate the evaluation, I will combine 2 good-relevant and ok-relevant volumes in the groundtruth into a single relevant set.

4.1. Metric

Here I use the mean Average Precision (MAP), a common metric used for evaluating the performance of a CBIR system. MAP measures the effectiveness of the system at retrieving relevant images from a database, based on a user's query.

Formula for calculating average precision:

$$AveP = \frac{\sum_{k=1}^{n} P(k)rel(k))}{numberofrelevantdocumentsintopk}$$

where $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise. Note that the average is over relevant documents in top-k retrieved documents and the relevant documents not retrieved get a precision score of zero.

Formula for calculating mean average precision:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

where Q is the number of queries.

The higher the mAP score, the better the performance of the CBIR system. MAP is a widely used metric for evaluating CBIR systems, as it takes into account the relevance of each retrieved image, and provides a comprehensive measure of the system's performance across multiple queries.

4.2. Comparision

I tested and compared systems with 4 different feature extractors based on mAP measurements, here are the results:

	Time (indexing)	Time (evaluate)	Time (retrieve)	mAP
RGBHistogram	113.915	3.285	0.968	0.923
LBP	277.980	4.728	1.252	0.886
Resnet50	143.658	6.802	2.136	0.972
VGG16	168.541	27.2301	3.154	0.938

Figure 10. Experiment

It can be seen that the model using Resnet50 has a very high efficiency. Processing time is also relatively fast.

5. Demo

The below results or demo are implemented circumspectly. I use Streamlit framework to build image retrieval system. With the input query is a image or any cropped window of that image, my expectaion is a set of relevant image from available database.

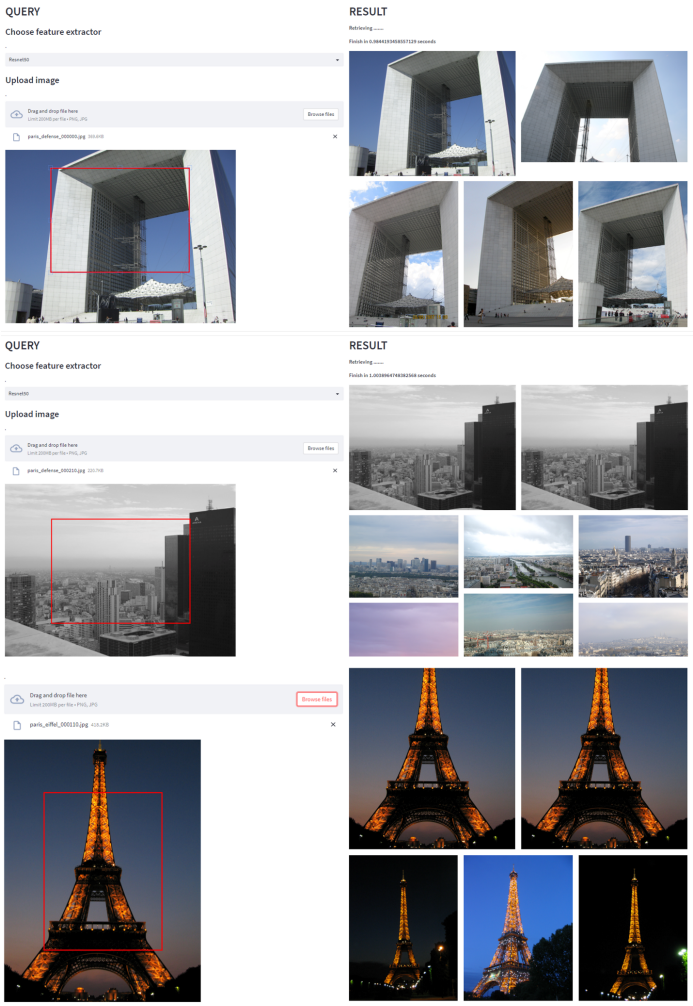

Figure 11. Results with system deployment

6. Conclusion

I have presented the goal of image retrieval, benchmark dataset contains 6K images search techniques on large image databases (state of the art method) is Faiss and comparing different methods. Content is reported and results is demonstrated. I try to research both of them to expect as possibly as the best project. Although there are some remained problems that I can not generalize above, as soon as the I can get a better hardware in the future, the input data and the algorithms are better.

I implement my algorithms and provide my source code on GitHub.[1] The programming language used is Python. Some libraries and tools are used:

- torchvision.models: use 2 models VGG16, Resnet (pretrained) to extract feature
  - pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html
  - pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html
- skimage.feature.local-binary-pattern: use LBP method to extract feature
  - https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html
- faiss: support indexing and search similar vector for the problem
  - https://github.com/facebookresearch/faiss
- streamlit: build system visualization interface
  - https://streamlit.io/
- streamlit-cropper: a streamlit custom component for easy image cropping
  - https://github.com/turner-anderson/streamlit-cropper

And some other libraries and tools to support reading file, loading data, building models, ...

[1] https://github.com/KhaLee2307/image-retrieval.git