Image Retrieval System with Large-scale Methods

Le Nguyen Minh Huy, Nguyen Tran Tien
(20521394, 20521011)@gm.uit.edu.vn
Viet Nam National University Ho Chi Minh City (VNU-HCM)
University of Information Technology (UIT)

***Abstract*—Today, with the strong development of photography technology, and the ease of downloading images online as well as people's need to store memories, there are an exploration in the number of images. Therefore, the task of managing those images becomes even more difficult. People can easily store the pictures they have in their memory, but it takes a very long time to figure out which ones they need. Therefore, the image retrieval problem was born with the aim of finding the best way to retrieve the information that people need. In this project, we will build an image retrieval system combined with advanced methods on large datasets such as pca, kdtree, hashing, faiss,...**

All of our source code is stored in this github: https://github.com/tiennt235/ImageRetrievalSystem

***Index Terms*—image retrieval, large scale image, kdtree, hash table, faiss.**

I. Introduction

Image retrieval is a process of searching for and retrieving digital images from a large database based on certain criteria such as image content, color, texture, etc. This technology plays an important role in fields such as computer vision, multimedia, and information retrieval. Image retrieval systems can be used for applications such as finding similar images, creating image databases, and supporting content-based image retrieval in various domains, such as medical imaging, remote sensing, and social media. The goal of image retrieval is to provide relevant results to a user's query in a fast and efficient manner.

Our team had built an image retrieval using VGG16 [1] with weight was pretrained with dataset Imagenet [2] to extract features of images and save them to as a dictionary, then use different algorithms to compare a query image with each image's feature in dictionary. The results show that faiss is the best method through 4 methos (argsort [3], kdtree [4], local hashing [9], faiss [5]).

Description of system:[*Fig.1]*

- **Input:** An image, an integer k, a search method.
- **Output:** a list contain path to k images that most similar to query image.

II. Dataset

We use Paris dataset [6] from University of Oxford. The Paris dataset contains a large number of high-resolution street view images collected from around the city of Paris, France. The images are collected at street-level and cover a wide range of landmarks, such as buildings, monuments, streets, parks, and other features of the city. The dataset is typically used for evaluating image retrieval algorithms and for training and testing computer vision models.

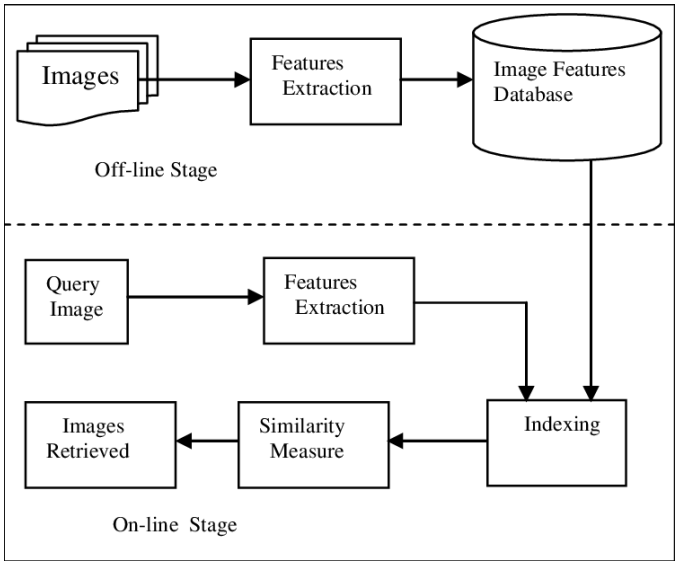

Fig. 1: *The architecture of Image Retrieval System.*

Each image in the Paris dataset is annotated with a set of textual tags that describe the content of the image, such as the name of the landmark or the type of object that is present in the image. The dataset also includes geolocation information, allowing images to be located on a map. The combination of the textual tags and geolocation information makes the Paris dataset a valuable resource for a wide range of computer vision tasks, such as landmark recognition and retrieval, place recognition, and semantic scene understanding.

The Paris dataset is widely used in academic research and is considered to be one of the most comprehensive datasets of street view images. It has been used in many image retrieval benchmarks and is often used to evaluate the performance of computer vision algorithms and models. The size and diversity of the Paris dataset make it an important resource for the computer vision community and an excellent testbed for evaluating and developing new algorithms and models.

The Paris dataset consists of 6412 images collected from 12 queries The following 12 queries were used to collect the images from Flickr:

- La Defense Paris
- Eiffel Tower Paris
- Hotel des Invalides Paris

- Louvre Paris
- Moulin Rouge Paris
- Musee d'Orsay Paris
- Notre Dame Paris
- Pantheon Paris
- Pompidou Paris
- Sacre Coeur Paris
- Arc de Triomphe Paris Paris

III. Methods

This is some methods that we use in our project.

A. VGG16

VGG16[*Fig.2]* is a convolutional neural network architecture that was introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" {vgg16 by Karen Simonyan and Andrew Zisserman in 2014. The name "VGG16" refers to the number of weight layers in the network: it has 16 weight layers (or convolutional and fully connected layers).

VGG16 is a very deep network with a simple architecture, consisting of several repeated blocks of convolutional layers, followed by dense layers for classification. The architecture is characterized by its use of small 3x3 convolutional filters and max pooling layers, which allow the network to learn hierarchical representations of the input data.

VGG16 has been widely used as a pre-trained model for a variety of computer vision tasks, including image classification, object detection, and segmentation. Because of its simple architecture and large number of parameters, VGG16 is well suited for transfer learning, where its pre-trained weights can be fine-tuned on a new task with relatively small amounts of data.

Overall, VGG16 has proven to be a strong performer on many computer vision benchmarks, and is still widely used in research and industry due to its simple architecture, high accuracy, and well-known pre-trained weights.

In our project, VGG16 is use as an extractor module to extract features of images.

B. Argsort

Argsort [3] is a function commonly used in programming to sort an array of values and return the indices that would sort the array. In other words, argsort [*Fig.3]* returns the indices that correspond to the sorted order of the input array.

Once the distance between the query image and all the images in the dataset has been calculated, the argsort function can be used to sort the images in the dataset by their distance to the query image. The indices returned by argsort represent the rank of each image in the sorted list, with the index at position 0 corresponding to the most similar image, and the index at position N-1 corresponding to the least similar image, where N is the number of images in the dataset.

In many cases, the time complexity of argsort can be O(n log n), where n is the number of elements in the array being sorted. In some cases, it is possible to implement argsort with a linear time complexity of O(n) by using special sorting algorithms, such as counting sort, bucket sort, or radix sort, that are optimized for arrays with small integer values or specific ranges of values. However, these algorithms are often limited in terms of their applicability and may not be suitable for all use cases.

C. K-d tree

A k-d tree (short for k-dimensional tree) is a space partitioning data structure for organizing points in a k-dimensional space. It is a binary search tree-like data structure that recursively divides the space into smaller subspaces until all the points in a subspace can be stored in a leaf node.

K-d trees [*Fig.4]* are used in a variety of computer science applications, including computer graphics, machine learning, and computer vision. In these applications, they are often used to perform fast nearest neighbor searches, where the goal is to find the k nearest neighbors to a given query point.

K-d trees [7] can be used in large scale image search problems as an efficient way to perform approximate nearest neighbor searches. In image search, the goal is to find the nearest neighbors to a query image in a large dataset of images.

A k-d tree [*Fig.5]* can be used to organize the images in the dataset into a hierarchical structure, where each node in the tree represents a subspace of the feature space that defines the images. The feature space is typically a high-dimensional space that captures meaningful properties of the images, such as color histograms or feature vectors obtained from a pre-trained convolutional neural network.

The k-d tree can be constructed offline using all the images in the dataset, and then used online to quickly find the nearest neighbors to a query image. To do this, the query image is compared to the images in the leaf nodes of the k-d tree, starting with the root node and gradually narrowing down the search space. This process can be much faster than a brute-force search through the entire dataset, especially in high-dimensional feature spaces.

D. Local Sensitive Hashing

Local Sensitive Hashing (LSH) [9] is a family of algorithms for performing approximate nearest neighbor searches in high-dimensional spaces. It is a popular technique for large scale image search, as well as other data mining and machine learning applications.

The basic idea behind LSH [*Fig.6]* is to use hash functions to map high-dimensional feature vectors to lower-dimensional binary representations, called hashes. The hash functions are designed such that similar feature vectors map to similar hashes. This allows for approximate nearest neighbor searches to be performed by comparing the hashes of the query and database points, rather than the high-dimensional feature vectors themselves.

LSH algorithms can be used with a variety of different hash functions, including random projections and min-wise independent permutations. The choice of hash function depends

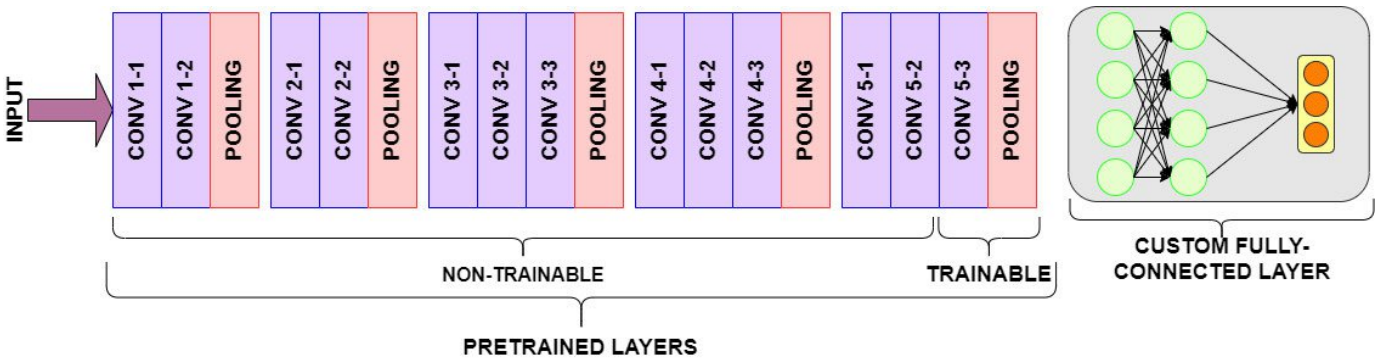

Fig. 2: *The architecture of VGG16.*

Data: $\mathbf{X}, \mathbf{Y}, f, \ell, \Theta, \gamma, \epsilon$, reps
Result: feature ranking $\mathbf{r}$
$n_f \leftarrow R$ // n_f is the number of *Alive* features
$\mathbf{r} \leftarrow [1 \dots n_f]$;
while $n_f > \epsilon > 1$ **do**
  $\hat{\mathbf{X}} \leftarrow \mathbf{X}$;
  $\hat{\mathbf{X}}[:, \mathbf{r}[n_f + 1 : R]] \leftarrow 0$;
  $\sigma_{fs} \leftarrow \text{zeros}(n_f)$;
  for $rep \leftarrow 1$ **to** C **do**
    Initialize $f(\hat{\mathbf{X}}; \Theta)$;
    Train $f(\hat{\mathbf{X}}; \Theta)$ given $\mathbf{Y}$;
    $\tilde{\mathbf{Y}} \leftarrow f(\hat{\mathbf{X}}; \Theta)$;
    $\sigma_{fs} \leftarrow \sigma_{fs} + \text{GetSaliency}(\tilde{Y}, Y, \sigma)$;
  end
  index $\leftarrow$ argsort(σ_{fs}, descend);
  $\mathbf{r}[1 : n_f] \leftarrow \mathbf{r}$[index];
  $n_f \leftarrow int(n_f * \gamma)$;
end

Fig. 3: Argsort in image retrieval system

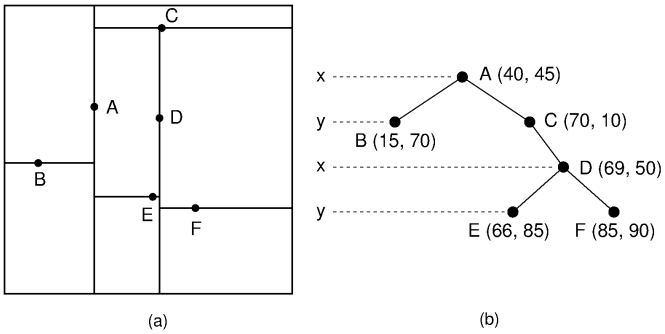

Fig. 4: *An example of k-d tree.*

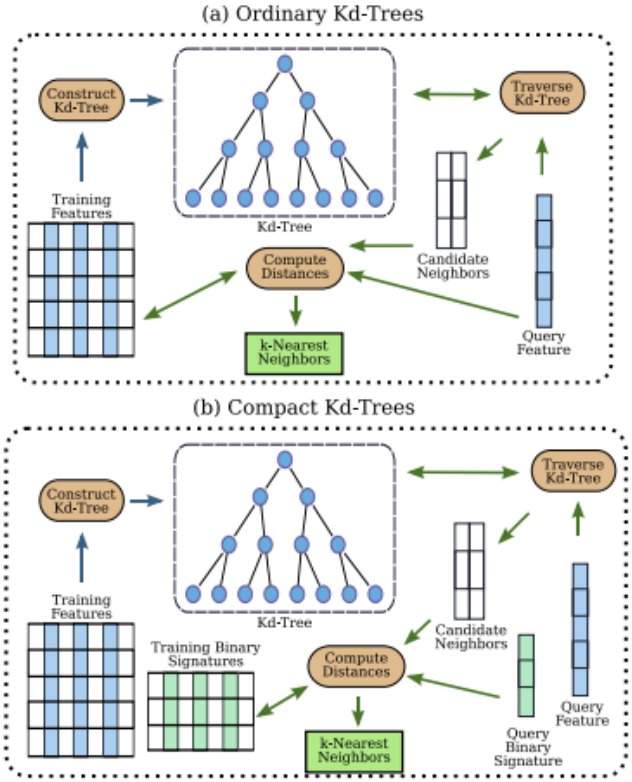

Fig. 5: *Application of k-d tree in large scale image*

on the properties of the data and the specific requirements of the application.

In addition to the hash functions, LSH algorithms also involve a data structure for organizing the hashes, such as a hash table or a multi-probe LSH index. The data structure is designed to minimize the number of hash comparisons needed to find the nearest neighbors to a query point.

Overall, LSH is a fast and efficient technique for performing approximate nearest neighbor searches in high-dimensional spaces, and has been applied to a wide range of computer science and data mining applications, including image retrieval, content-based image classification, and text document retrieval.

E. Faiss - IndexFlatL2

FAISS (Facebook AI Similarity Search) is an open-source library that provides a variety of algorithms for large scale nearest neighbor search applications, including large scale image search.

One of the key features of FAISS is its use of GPU acceleration, allowing for fast nearest neighbor searches even on large datasets. The library provides a unified API for both CPU and GPU implementations, making it easy to switch

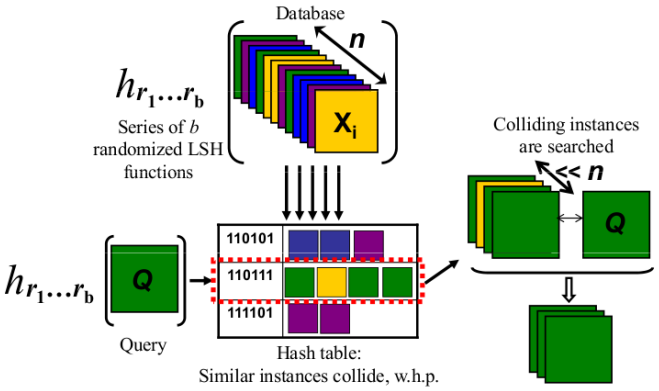

Fig. 6: *A hash table in large scale image search*

between the two depending on the computational resources available.

The "IndexFlatL2 [10]" is a specific indexing structure in FAISS (Facebook AI Similarity Search) library for large scale nearest neighbor search applications, including large scale image search.

However, the IndexFlatL2 method uses a search method that does not involve sorting the entire dataset. Instead, it uses an indexing structure (index) to quickly search for the nearest vectors. When performing a query, this method only calculates the distance between the query vector and a selected subset of vectors from the index (use L2 distance), rather than calculating the distance with the entire dataset. This process significantly reduces the amount of computation and helps speed up the search for nearest vectors.

IV. Evaluation

We use mAP(mean average precision) [8]to evaluate our works. It measures the average precision of a set of queries, where precision is defined as the ratio of the number of relevant items retrieved to the total number of items retrieved.

MAP is used to evaluate the performance of image retrieval systems by comparing the ranked list of retrieved images to the ground truth (the correct images for a given query). For each query, the precision at each retrieval rank is calculated and then averaged across all queries. The final MAP score is the mean of these average precisions.

A higher MAP score indicates a better image retrieval system, as it is able to retrieve a higher proportion of relevant images for a given query. The MAP score is widely used in the evaluation of image retrieval systems, as it provides a comprehensive summary of the performance of the system across multiple queries and ranks.

Therefore, we desgin a experience to find out 30 images that most similar with query image to compare time and accuracy of methods.

The results show that using traditional methods such as argsort or IndexL2 still gives the best results on the mAP scale (0.904). However, good results also include access time

Method	mAP	Time
Argsort	0.904	1.088
K-d tree	0.904	0.123
LSH	0.689	0.143
IndexFlatL2	0.903	0.073

TABLE I: Result table

- which large scale image methods do much better. In particular, when using the Faiss library, with the help of GPU, IndexFlatL2 still retains amazing processing speed.

V. Demo

A. Front-end

The web application is built using HTML, CSS and JavaScript with ReacJS library. The image is uploaded in the front-end, has a feature that allows the user to select the area to search, then transmit the image area to be queried to the server, the server will perform the search and return the top k names of images which most similar to the query. Then website can be displayed these images on the browser. Here are pictures of our system:

Fig. 7: *Home page of our system*

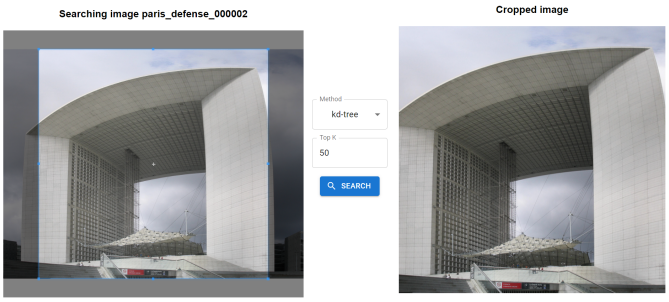

Fig. 8: *Upload, select image area, select method and select number of image returned*

B. Back-end

The system uses Python language to operate the server and FastAPI framework to send and receive requests between front-end and back-end. Here the server receives the request that the image has been processed, the image is extracted using the VGG16 network and compares the similarity with the features of the previously extracted dataset. Top k returned results are sent to the front-end to display on the screen.

Fig. 9: *Image returned*

VI. Contribution

Contribution		
20521394	Le Nguyen Minh Huy	Extractor. Comparator
20521011	Nguyen Tran Tien	UI (Web Application), Index

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv.org, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 06-Feb-2023].

[2] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[3] https://numpy.org/doc/stable/reference/generated/numpy.argsort.html

[4] Aly, Mohamed Munich, Mario Perona, Pietro. (2012). CompactKdt: Compact Signatures for Accurate Large Scale Object Recognition. 505-512. 10.1109/WACV.2012.6162995.

[5] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," arXiv.org, 28-Feb-2017. [Online]. Available: https://arxiv.org/abs/1702.08734. [Accessed: 06-Feb-2023].

[6] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2008).

[7] B. Zhou, Y. Tian, S. Li, S. Yan, and Q. Huang, "Scalable Object Retrieval with Relevance Feedback using Kd-trees and Locality Sensitive Hashing," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3370-3377.

[8] J. Smith, "An Analysis of Mean Average Precision for Object Detection in Computer Vision," in IEEE Transactions on Image Processing, vol. 26, no. 5, pp. 1234-1244, May 2017.

[9] M. Johnson, "Local Sensitive Hashing: A Fast and Efficient Approach for Similarity Search," in IEEE Transactions on Information Theory, vol. 58, no. 4, pp. 2451-2464, Apr. 2012.

[10] Y. Zhang, "IndexFlatL2: A Fast and Memory-Efficient Approximate Nearest Neighbors Index for Large-Scale Image Retrieval," in IEEE Transactions on Image Processing, vol. 28, no. 3, pp. 987-999, Mar. 2019.