VIETNAM NATIONAL UNIVERSITY HO CHI MINH

UNIVERSITY OF INFORMATION TECHNOLOGY

Final Project:

IMAGE RETRIEVAL

Subject: Multimedia Information Retrieval

Class: CS336.M11.KHCL

Lecturer: Ngo Duc Thanh

Students:

• Thai Tran Khanh Nguyen	- 19520188
• Doan Nguyen Nhat Quang	- 19520235
• Nguyen Pham Vinh Nguyen	- 19520186
• Nguyen Khanh Nhu	- 19520209

Ho Chi Minh, January 2022

Table of Contents

1. Introduction:

Image Retrieval is a fundamental task in computer vision, since it is directly related to various practical applications such as object detection, visual place recognition and product recognition. The last decades have witnessed tremendous advances in image retrieval systems - from handcrafted features and indexing algorithms to, more recently, methods based on convolutional neural networks (CNNs) for global descriptor learning. With the growth of image retrieval systems as well as e-commerce and online websites, image retrieval applications have been increasingly all along around our daily life. For example, Amazon, Alibaba, Myntra etc. have been heavily utilizing image retrieval to put forward what they think is the most suitable product based on what we have seen just now.

In this report, we conduct research and experiment with different methods to build Image Retrieval system. First, the Simple Image Retrieval (SIR) method uses an existing CNN network architecture to extract features of the input images and compare the feature vectors by Cosine Distance. For the other two methods, we examine state of the art methods with the Oxford5k dataset, and the source code is publicly available on github. Includes Deep Local Feature (DELF) and CNN Image Retrieval with No Human Annotation - an unsupervised learning method.

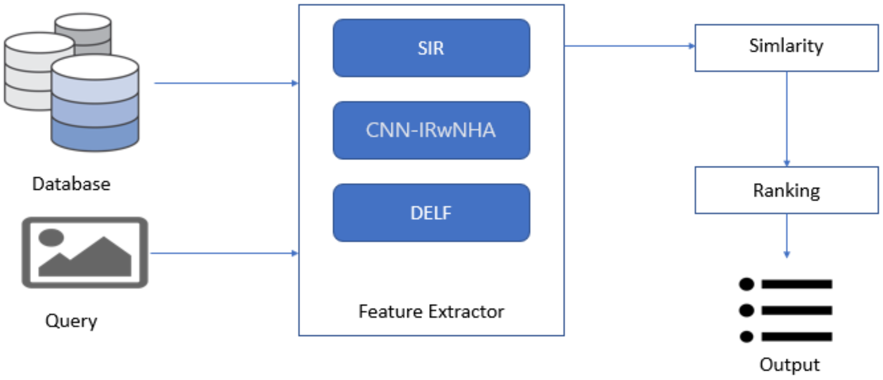
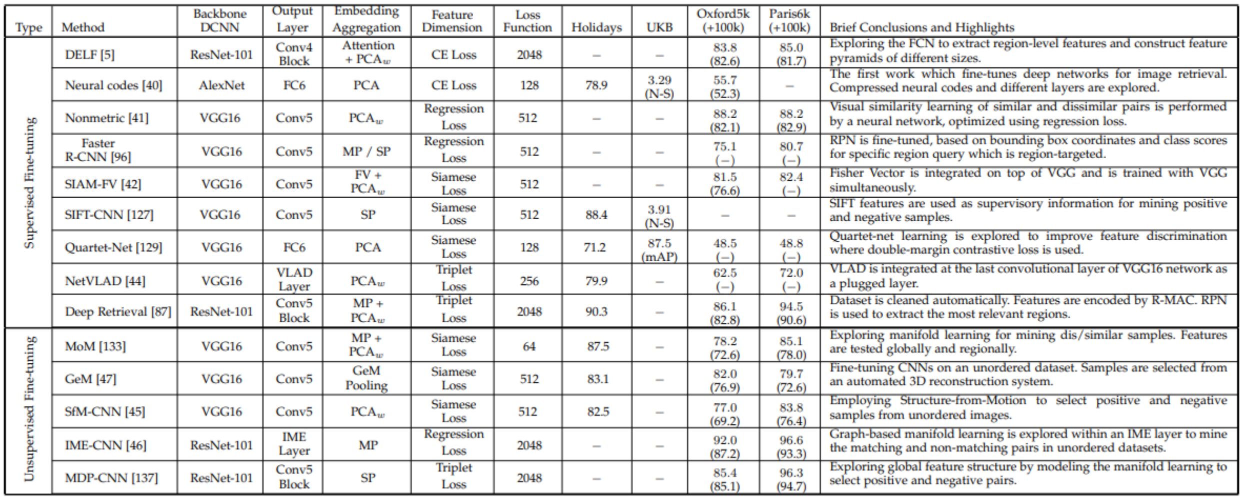


Figure 1: Our IR System Architecture

After being designed, these methods are evaluated by us against the ground truth of the Oxford5k dataset. We use the map to draw conclusions about the pros and cons of the methods in finding relevant images. Finally, we deploy our system to the web application. Because of limited resources, our demo version is temporarily hosted by google colab for each demo run.

2. Survey:

Before building our Image Retrieval System, we started surveying many sources. First of all, we searched on Google with the keyword “image retrieval” to have an overview of it. Secondly, we started looking at some science papers to understand how image retrieval systems work, especially the systems used methods based on convolutional neural networks (CNNs). The Deep Learning for Instance Retrieval: A Survey [1] scientific paper has been surveyed by us to find the best current methods. Some reputable websites in this field such as paperwithcode were also investigated by us during the survey. Finally, we started looking for the source code of the systems choosing from science papers based on what we desired, then tried executing them.



Type	Method	Backbone DCNN	Output Layer	Embedding Aggregation	Feature Dimension	Loss Function	Holidays	UKB	Oxford5k (+100k)	Paris6k (+100k)	Brief Conclusions and Highlights
Supervised Fine-tuning	DELF [5]	ResNet-101	Conv4 Block	Attention + PCA_w	CE Loss	2048	–	–	83.8 (82.6)	85.0 (81.7)	Exploring the FCN to extract region-level features and construct feature pyramids of different sizes.
	Neural codes [40]	AlexNet	FC6	PCA	CE Loss	128	78.9	3.29 (N-S)	55.7 (52.3)	–	The first work which fine-tunes deep networks for image retrieval. Compressed neural codes and different layers are explored.
	Nonmetric [41]	VGG16	Conv5	PCA_w	Regression Loss	512	–	–	88.2 (82.1)	88.2 (82.9)	Visual similarity learning of similar and dissimilar pairs is performed by a neural network, optimized using regression loss.
	Faster R-CNN [96]	VGG16	Conv5	MP / SP	Regression Loss	512	–	–	75.1 (–)	80.7 (–)	RPN is fine-tuned, based on bounding box coordinates and class scores for specific region query which is region-targeted.
	SIAM-FV [42]	VGG16	Conv5	FV + PCA_w	Siamese Loss	512	–	–	81.5 (76.6)	82.4 (–)	Fisher Vector is integrated on top of VGG and is trained with VGG simultaneously.
	SIFT-CNN [127]	VGG16	Conv5	SP	Siamese Loss	512	88.4	3.91 (N-S)	–	–	SIFT features are used as supervisory information for mining positive and negative samples.
	Quartet-Net [129]	VGG16	FC6	PCA	Siamese Loss	128	71.2	87.5 (mAP)	48.5 (–)	48.8 (–)	Quartet-net learning is explored to improve feature discrimination where double-margin contrastive loss is used.
	NetVLAD [44]	VGG16	VLAD Layer	PCA_w	Triplet Loss	256	79.9	–	62.5 (–)	72.0 (–)	VLAD is integrated at the last convolutional layer of VGG16 network as a plugged layer.
	Deep Retrieval [87]	ResNet-101	Conv5 Block	MP + PCA_w	Triplet Loss	2048	90.3	–	86.1 (82.8)	94.5 (90.6)	Dataset is cleaned automatically. Features are encoded by R-MAC. RPN is used to extract the most relevant regions.
Unsupervised Fine-tuning	MoM [133]	VGG16	Conv5	MP + PCA_w	Siamese Loss	64	87.5	–	78.2 (72.6)	85.1 (78.0)	Exploring manifold learning for mining dis/similar samples. Features are tested globally and regionally.
	GeM [47]	VGG16	Conv5	GeM Pooling	Siamese Loss	512	83.1	–	82.0 (76.9)	79.7 (72.6)	Fine-tuning CNNs on an unordered dataset. Samples are selected from an automated 3D reconstruction system.
	SfM-CNN [45]	VGG16	Conv5	PCA_w	Siamese Loss	512	82.5	–	77.0 (69.2)	83.8 (76.4)	Employing Structure-from-Motion to select positive and negative samples from unordered images.
	IME-CNN [46]	ResNet-101	IME Layer	MP	Regression Loss	2048	–	–	92.0 (87.2)	96.6 (93.3)	Graph-based manifold learning is explored within an IME layer to mine the matching and non-matching pairs in unordered datasets.
	MDP-CNN [137]	ResNet-101	Conv5 Block	SP	Triplet Loss	2048	–	–	85.4 (85.1)	96.3 (94.7)	Exploring global feature structure by modeling the manifold learning to select positive and negative pairs.

Figure 2: Table of Performance of Evaluation of methods

Afterall, we have chosen three methods which meet our needs. The first method is based on CNN using the ResNet152 which used like backbone of our method SIR. In order to increase generalizability and empiric various methods, our team decided to choose one of the most popular supervised and unsupervised methods. With the supervised method, we find the DELF method [2] which has the third highest score among the methods of its kind. In addition, this method has official source code on tensor flow which will be easy to implement. With the unsupervised one, we found the article Fine-tuning CNN Image Retrieval with No Human Annotation [3].

3. Method and Model:

3.1 Simple Image Retrieval (SIR):

3.1.1 Architecture and Transfer:

For this approach, we use Resnet152 network architecture for input feature extraction. We remove the last layer of Resnet152, then connect the last flatten layer with a fully connected layer with a unit number of 1024 (this is also the dimensionality of each feature vector after it has been extracted). To increase the accuracy of this model, we freeze all previous parameters and conduct training on later classes added with the IMAGENET dataset.

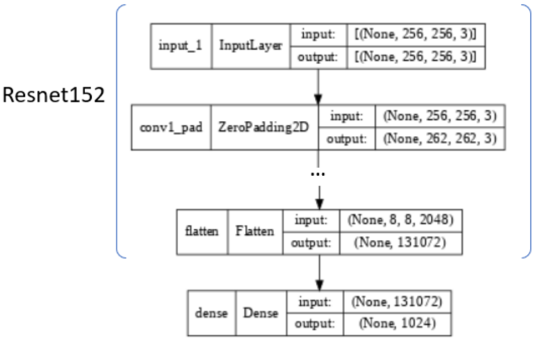

Resnet152
input_1 | InputLayer | input: | [(None, 256, 256, 3)]
output: | [(None, 256, 256, 3)]
conv1_pad | ZeroPadding2D | input: | (None, 256, 256, 3)
output: | (None, 262, 262, 3)
...
flatten | Flatten | input: | (None, 8, 8, 2048)
output: | (None, 131072)
dense | Dense | input: | (None, 131072)
output: | (None, 1024)

Figure 3: Architecture SIR

3.1.2 Similarity:

We use Cosine Similarity to compare the similarity of feature vectors. The formula is defined as follows:

$$\text{cosine\_similarity}(\boldsymbol{A},\boldsymbol{B}) = \frac{\boldsymbol{A}\,.\,\boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{\mathrm{i}}^{\mathrm{n}} \boldsymbol{A}_i \boldsymbol{B}_i}{\sqrt{\sum_{\mathrm{i}}^{\mathrm{n}} \boldsymbol{A}_i^2}\sqrt{\sum_{\mathrm{i}}^{\mathrm{n}} \boldsymbol{B}_i^2}} \qquad (1)$$

3.2 Deep Local Feature (DELF):

DELF (DEep Local Feature) is an attentive local feature descriptor suitable for large-scale image retrieval. The new feature is based on convolutional neural networks, which

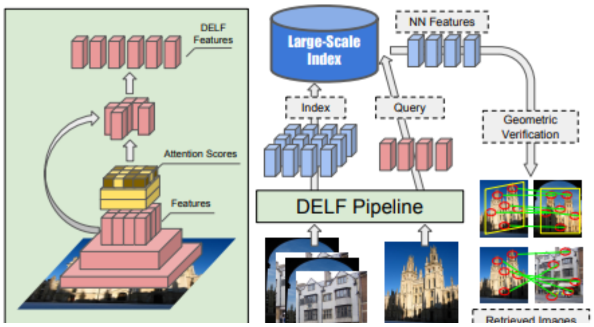


Figure 4: Overall architecture use DELF

are trained only with image-level annotations on a landmark image dataset. To identify semantically useful local features for image retrieval, we also propose an attention mechanism for keypoint selection, which shares most network layers with the descriptor.

3.2.1 Feature Extraction:

In this stage, DELF extracts dense features from an image by applying a fully convolutional network (FCN), which is constructed by using the feature extraction layer of CNN trained with a classification loss. The FCN is taken from the ResNet50 model, using the output of the conv4_x convolutional block. To handle scale changes, DELF is explicitly constructed an image pyramid and applied the FCN for each level independently. The obtained feature maps are regarded as a dense grid of local descriptors. Features are localized based on their receptive fields, which can be computed by considering the configuration of convolutional and pooling layers of the FCN. DELF uses the pixel coordinates of the center of the receptive field as the feature location. The receptive field size for the image at the original scale is 291 × 291. Using the image pyramid, DELF can obtain features that describe image regions of different sizes.

3.2.2 Keypoint Selection:

After extracting features, DELF has a technique to effectively select a subset of the features. Since a substantial part of the densely extracted features are irrelevant to the recognition module and likely to add clutter, distracting the retrieval process, keypoint selection is important for both accuracy and computational efficiency of retrieval systems.

The keypoint selection needs to train a landmark classifier with attention to explicitly measure relevance scores for local feature descriptors. To train the function, features are pooled by a weighted sum, where the weights are predicted by the attention network. The training is formulated as follows. Denote by $\mathbf{f}_\mathrm{n} \in R^d, n = 1, \ldots, N$ the d-dimensional features to be learned jointly with the attention model. Our goal is to learn a score function $\alpha(\mathbf{f}_\mathrm{n};\theta)$ for each feature, where θ denotes the parameters of function $\alpha(\cdot)$. The output logit y of the network is generated by a weighted sum of the feature vectors, which is given by

$$y = \mathbf{W}\,(\sum_{\mathrm{n}} \alpha(\mathbf{f}_\mathrm{n};\theta)\,.\,\mathbf{f}_\mathrm{n} \qquad (2)$$

where $\mathbf{W} \in \boldsymbol{R}^{M\,x\,d}$ represents the weights of the final fully connected layer of the CNN trained to predict M classes. For training, using cross entropy loss, which is given by

$$\mathcal{L} = -y*.\,log(\frac{exp(\boldsymbol{y})}{\mathbf{1}^T exp(\boldsymbol{y})}) \qquad (3)$$

where y* is ground-truth in one-hot representation and (1) is one vector. The parameters in the score function $\alpha(\cdot)$ are trained by backpropagation, where the gradient is given by

$$\frac{\partial\mathcal{L}}{\partial\theta} = \frac{\partial\mathcal{L}}{\partial\mathbf{y}}\sum_{n}\frac{\partial\boldsymbol{y}}{\partial\alpha_n}\frac{\partial\alpha_n}{\partial\theta} = \frac{\partial\mathcal{L}}{\partial\mathbf{y}}\sum_{n}\boldsymbol{W}\,\mathbf{f}_\mathrm{n}\frac{\partial\alpha_n}{\partial\theta} \qquad (4)$$

where the backpropagation of the output score $\alpha_n \equiv \alpha(\mathbf{f}_n; \theta)$ with respect to θ is same as the standard multi-layer perceptron.

Restricting α(·) to be non-negative, to prevent it from learning negative weighting. The score function is designed using a 2-layer CNN with a softplus activation at the top. For simplicity, we employ the convolutional filters of size 1×1, which work well in practice. Once the attention model is trained, it can be used to assess the relevance of features extracted by our model.

3.2.3 Dimensionality Reduction:

Reducing the dimensionality of selected features to obtain improved retrieval accuracy. First, the selected features are L2 normalized, and their dimensionality is reduced to 40 by PCA, which presents a good trade-off between compactness and discriminativeness. Finally, the features once again undergo L2 normalization.

3.3 CNN Image Retrieval with No Human Annotation (CNN-IRwNHA):

3.3.1 Fully Convolutional Network:

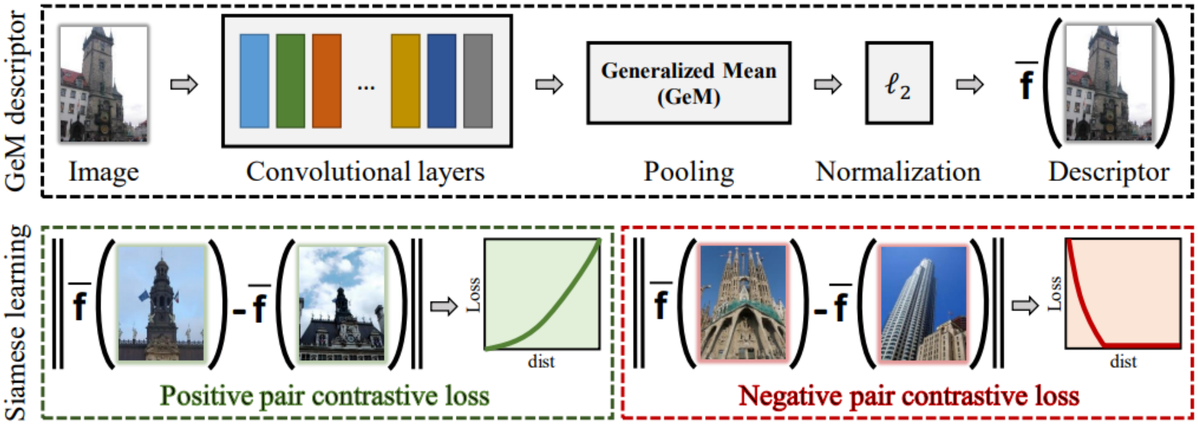


Figure 5: The architecture of network with the contrastive loss used at training time.

At this stage, given an input image, the output is a 3D tensor $\mathbf{X}$ of $W \times H \times K$ dimensions where K is the number of feature maps in the last layer. Let X_k be the set of W × H activations for the feature map $k \in \{1 \ldots K\}$. The network output consists of K such activation sets or 2D feature maps.

3.3.2 Generalized-mean pooling and image descriptor:

At the pooling process, taken 3D tensor X as an input and produced a vector f as an output. The vector f is produced by using generalize-mean pooling whose result is

$$\mathbf{f}^{(m)} = [f_1^{(m)} \ldots f_k^{(m)} \ldots f_K^{(m)}]^T, \quad f_k^{(m)} = \max_{x \in X_k} x, \qquad (5)$$

The feature vector finally consists of a single value per feature map, the generalized-mean activation, and its dimensionality is equal to K. The pooling parameter pk can be manually set or learned since this operation is differentiable and can be part of the backpropagation. There is a different pooling parameter per feature map $\mathbf{f}$ but it is also possible to use a shared one. In this case $p_k = p, \forall k \in [1, K]$. The last network layer comprises an L2-normalization layer. Vector $\mathbf{f}$ is L2-normalized so that similarity between two images is finally evaluated with inner product. In the rest of the paper, GeM vector corresponds to the L2- normalized vector $\mathbf{f}$ and constitutes the image descriptor.

3.3.3 Siamese learning and loss function:

At the training stage, use a siamese architecture and train a two-branch network. Each branch is a clone of the other, meaning that they share the same parameters. The training input consists of image pairs (i, j) and labels $\mathbf{Y}\,(i, j) \in \{0, 1\}$ declaring whether a pair is non-matching (label 0) or matching (label 1). We employ the contrastive loss that acts on matching and non-matching pairs and is defined as

$$\mathcal{L}(i,j) = \begin{cases} \frac{1}{2}\left\|\bar{\mathbf{f}}(i) - \bar{\mathbf{f}}(j)\right\|^2, if\ \boldsymbol{Y}(i,j) = 1 \\ \frac{1}{2}(max\{0, \tau - \left\|\bar{\mathbf{f}}(i) - \bar{\mathbf{f}}(j)\right\|^2\}), if\ \boldsymbol{Y}(i,j) = 0 \end{cases} \tag{6}$$

where $\bar{\mathbf{f}}(i)$ is the L2-normalized GeM vector of image i, and τ is a margin parameter defining when non-matching pairs have large enough distance in order to be ignored by the loss.

4. Evaluation:

Roxford 5k dataset (Revisiting Oxford) [4]:

Author revisits and address issues with Oxford 5k and Paris 6k image retrieval benchmarks. New annotation for both datasets is created with an extra attention to the reliability of the ground truth and three new protocols of varying difficulty are introduced. Author additionally introduces 15 new challenging queries per dataset and a new set of 1M hard distractors.

Method	**map**			**map@5**		
	map E	map M	map H	map E	map M	map H
SIR	14.4	11.85	2.34	32.86	27.71	24.86
DELF	58.56	42.04	16.98	84.29	73.86	66.57
CNN-IRwNHA	85.08	68.65	44.24	92.86	92.29	86.

Table 1: Performance of our methods in roxford5k

Oxford 5k dataset:

Method	**map**
SIR	23.34
DELF	66.69
CNN-IRwNHA	82.09

Table 2: Performance of our methods in oxford5k

Considering the two tables of evaluation results, the best method for our IR system is CNN-IRwNHA. For the SIR method, we call this a naive method because it simply uses CNN architectures to extract global features. This leads to some local features that are not extracted well by the other two methods. In addition, we perform training on the IMAGENET dataset, which is a dataset used for object detection and classification, whose features are not suitable for the retrieval problem. DELF improves accuracy but the execution time for a rather large query (~2min/query) slows down the IR system. CNN-IRwNHA achieved the best results in our experiments in both accuracy and time.

5. Design API:

In this project, we build an API using method RESTful API Flask framework for server-side and ReactJS framework for client-side.

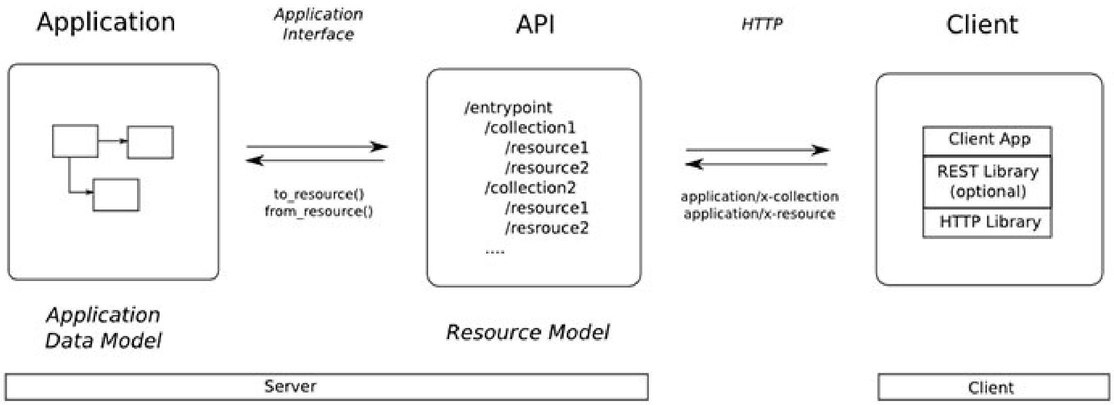


Figure 6: RESTful API

Here is our API workflow:

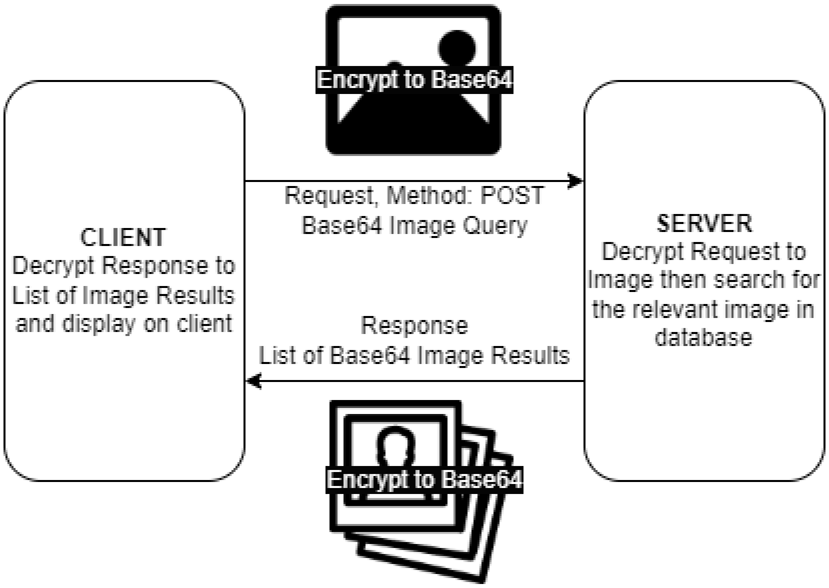

Encrypt to Base64
CLIENT
Decrypt Response to List of Image Results and display on client
Request, Method: POST
Base64 Image Query
Response
List of Base64 Image Results
SERVER
Decrypt Request to Image then search for the relevant image in database
Encrypt to Base64

Figure 7: API workflow

However, we cannot deploy both back-end and front-end on any free hosting server because our models require GPU for feature extraction step. Therefore, we use Google Colab (GPU available for free plan account) for hosting the back-end server then we just paste the link to the client website which has been deploy already to connect with the back-end server.

Experimental images

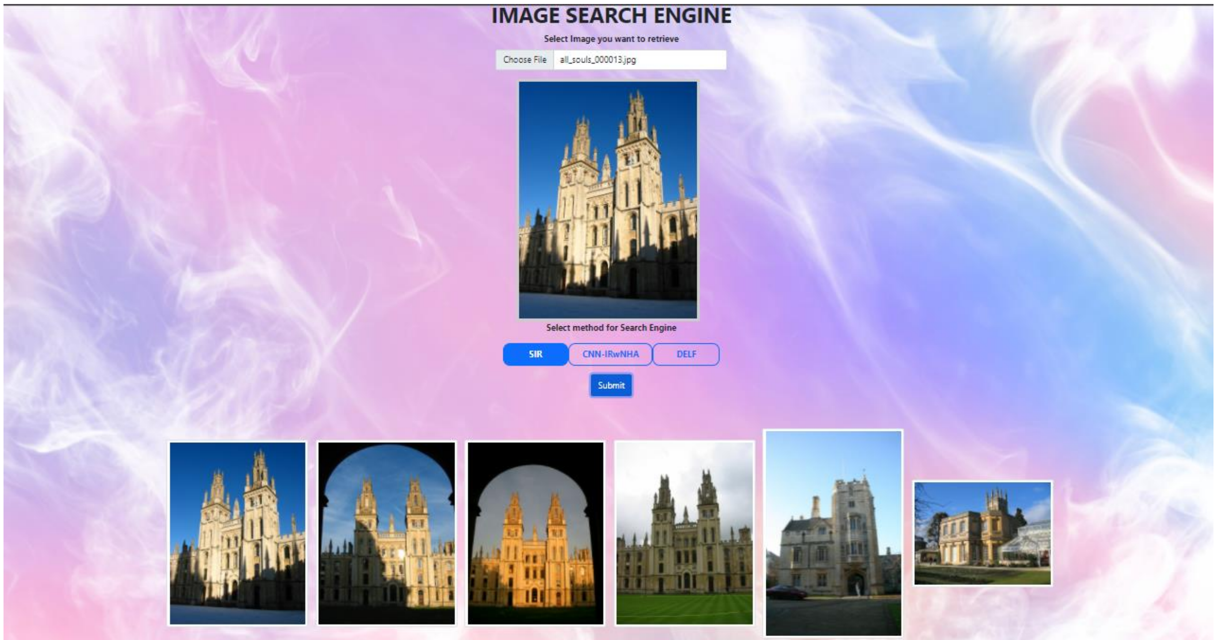


IMAGE SEARCH ENGINE
Select Image you want to retrieve
Select method for Search Engine
SIR
CNN-IRwNHA
DELF
Submit

Choose File all_souls_000013.jpg

Figure 8: Demo image

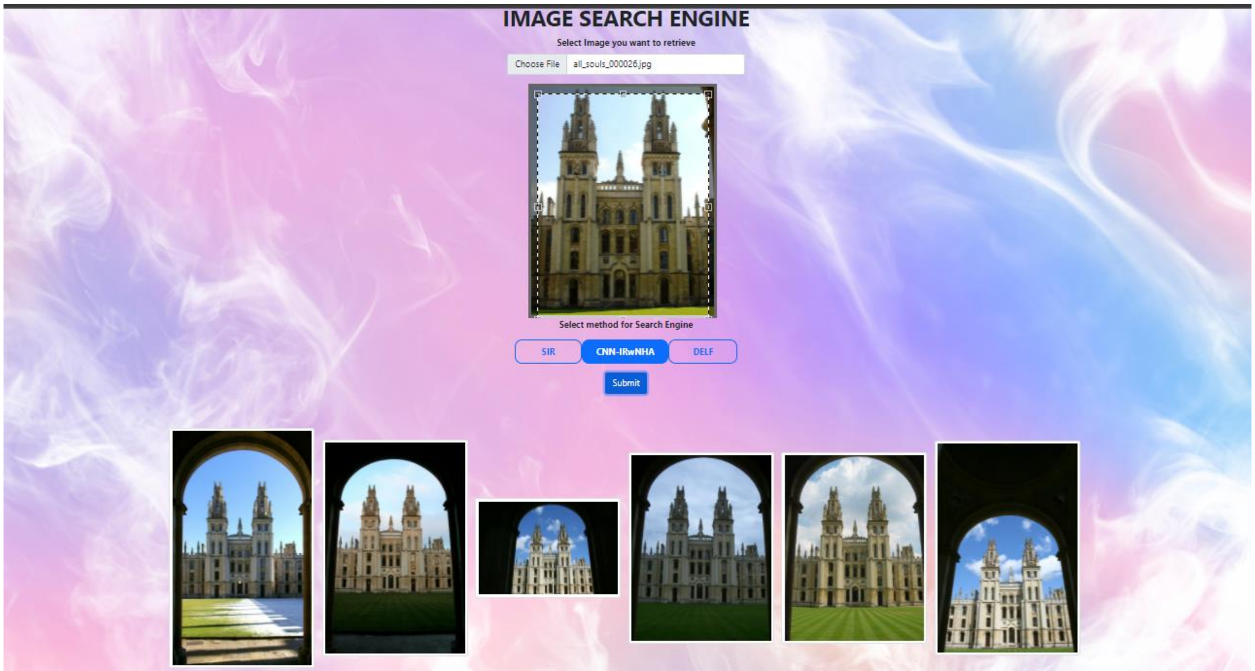


IMAGE SEARCH ENGINE
Select Image you want to retrieve
Select method for Search Engine
SIR
CNN-IRwNHA
DELF
Submit

Choose File all_souls_000026.jpg

Figure 9: Demo image

6. Conclusion:

Indeed, we have completely designed our Image Retrieval system and evaluated three methods that we used to build it. In practice, the unsupervised method CNN-IRwNHA have demonstrated the best performance. In term of using the unsupervised method, we figured out that this method could fine tune parameters itself to improve the performance of our Image Retrieval system, which helps us a lot for not spending too much time to find out how to upgrade our system. In contrast, the other methods need more time to fine tune parameters, however, their performances are still really good such as … for SIR method and … DELF method respectively.

When building the API, we have struggled with deploying our system on the host server, so we decided to build it as a web app. In this report, we provide a file called "Demo_FinalProject.ipynb" including detailed instructions for using our Image Retrieval system. However, our system can be used for Google Colab. The video below including instructions to use our system and our experiments.
(Link video: https://youtu.be/HQFgYrPgjX4)

7. References:

[1] Wei Chen, Yu Liu, Weiping Wang, Erwin M. Bakker, Theodoros Georgiou, Paul Fieguth, Li Liu, and Michael S. Lew, "Deep Learning for Instance Retrieval: A Survey", 2022

[2] H. Noh, A. Araujo, J. Sim, T. Weyand and B. Han, "Large-Scale Image Retrieval with Attentive Deep Local Features", ICCV, 2017

[3] Filip Radenovic, Giorgos Tolias, Ondrej Chum, "Fine-tuning CNN Image Retrieval with No Human Annotation", TPAMI, 2018

[4] F. and Iscen, A. and Tolias, G. and Avrithis, Y. and Chum, "Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking", CVPR, 2018

End