NATIONAL UNIVERSITY OF HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

COMPUTER SCIENCE

PROJECT REPORT

SUBJECTS: MULTIMEDIA INFORMATION RETRIEVAL

TOPIC:

Image Retrieval with Deep Learning

Class: CS336.N11.KHCL

Instructor: Dr. Ngo Duc Thanh

Students:

Le Tan Loc	20521546
Pham Hoang Phuc	20520278
Nguyen Hong Anh Thu	20520313
Huynh Nguyen Van Khanh	20521446

Ho Chi Minh City, February 20, 2023

Contents

I. Overview

1. Introduction about image retrieval model

Recent years have seen a rapid increase in the size of digital image collections. Everyday, both military and civilian equipment generates giga-bytes of images. A huge amount of information is out there. However, we cannot access or make use of the information unless it is organized so as to allow efficient browsing, searching, and retrieval. Image retrieval has been a very active research area since the 1970s, with the thrust from two major research communities, database management and computer vision. These two research communities study image retrieval from different angles, one being text-based and the other visual-based.

The text-based image retrieval can be traced back to the late 1970s. A very popular framework of image retrieval then was to first annotate the images by text and then use text-based database management systems (DBMS) to perform image retrieval. Many advances, such as data modeling, multidimensional indexing, and query evaluation, have been made along this research direction. However, there exist two major difficulties, especially when the size of image collections is large (tens or hundreds of thousands). One is the vast amount of labor required in manual image annotation. The other difficulty, which is more essential, results from the rich content in the images and the subjectivity of human perception. That is, for the same image content different people may perceive it differently. The perception subjectivity and annotation impreciseness may cause unrecoverable mismatches in later retrieval processes.

In the early 1990s, because of the emergence of large-scale image collections, the two difficulties faced by the manual annotation approach became more and more acute. To overcome these difficulties, content-based image retrieval was proposed. That is, instead of being manually annotated by text-based key words, images would be indexed by their own visual content, such as color and texture. Since then, many techniques in this research direction have been developed and many image retrieval systems, both research and commercial, have been built.

In currently modern society, we may be easy to see the great role of image retrieval systems based on content-based image retrieval (*CBIR*). It is known as query by image content, is the application of Computer Vision techniques to the Image Retrieval problem. With the application of image search techniques, the difficulty in representing images or graphic designs, etc. into queries has been solved, and they have been widely applied in systems such as Google, Lazada, Shopee, ...

2. Goal of the project

In this project, our team will research and learn about the processes for building a CBIR system and develop a CBIR system which is received one image (maybe full or partial photo) as the input query, and returns the top most relevant images, meeting the user's information needs.

The system is developed on *The Oxford Buildings* and *The Paris* dataset with some state-of-the-art models such as *VGG16, Xception, EfficientNetV2L*. *Mean Average Precision (mAP)* measure will be utilized to evaluate and compare between methods. After all, we will deploy system on the website in order to serve user's image search demands with friendly interface and simple to use.

3. Dataset description

We will use two datasets provided by *Visual Geometry Group* from University of Oxford: *The Oxford Buildings Dataset* and *The Paris Building Dataset*. Both datasets include a directory of photos and a directory of ground truth files.

3.1. The Oxford Building Dataset

The Oxford Buildings Dataset consists of 5062 images collected from *Flickr* by searching for particular Oxford landmarks. The collection has been manually annotated to generate a comprehensive ground truth for 11 different landmarks, each represented by 5 possible queries. This gives a set of 55 queries over which an object retrieval system can be evaluated. For each image and landmark in dataset, one of four possible labels was generated:

- Good - A nice, clear picture of the object/building.
- OK - More than 25% of the object is clearly visible.
- Bad - The object is not present.
- Junk - Less than 25% of the object is visible, or there are very high levels of occlusion or distortion.

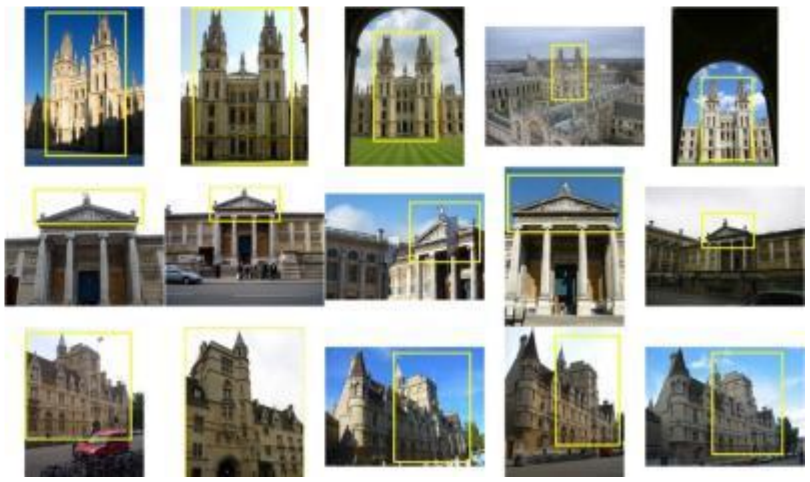


Figure 1 The Oxford Building Dataset

3.2. The Paris Building Dataset

Similar to *The Oxford Buildings Dataset*, *The Paris Building Dataset* has 6412 high-resolution images of buildings in different neighborhoods throughout the city of Paris collected from *Flickr* by searching for particular Paris landmarks. The images are available in JPEG format and are organized into folders according to their neighborhood or arrondissement.

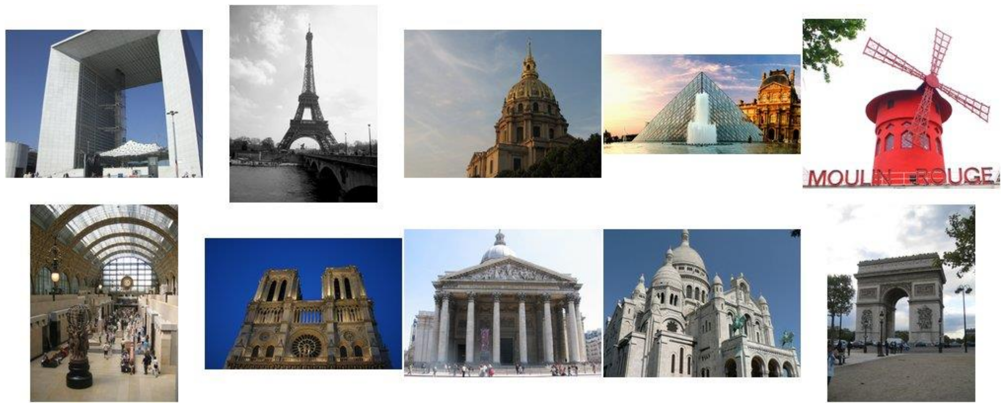
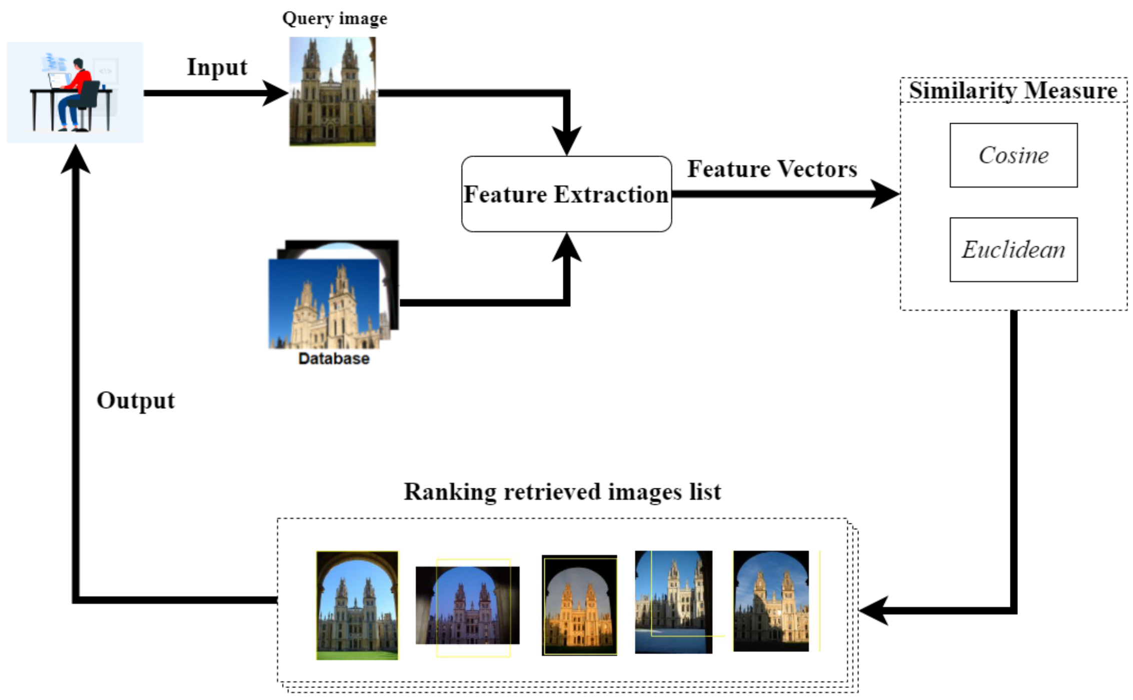


Figure 2 The Paris Building Dataset

II. Approach to the problem

1. Workflow



By using input image as query, we perform feature extraction to convert it into a feature vector. The popular method is to employ the simple Bag-of-Visual-Words methodology, which is both accessible and understandable. A recently developed and effective method, based on a deep learning model. In this project, our team will develop image retrieval system with 6 deep learning models namely: *VGG-16*, *Xception*, *InceptionV3*, *InceptionResNetV2*, *ResNet152V2*, *EfficientNetV2L* and proceed to evaluate and compare them; however, when deploying system on the website, we will only use the most effective model (*Xception*).

The system handles the image retrieval problem with:

+ Input: A image (maybe full or partial picture).

+ Output: List of ranked relevant images.

Regarding to database for system, we will preprocess and use VGG-16 network to collect vector features and synthesize these vectors into a vector and stored.

For each query image, we perform the same feature extraction steps.

After obtaining the feature vector, calculate the similarity between the representation vector of the query image and the vector representing the image in the database.

The result returns to users that is ranking list of high similarity sorted descending order.

2. Handling details

2.1. Image feature extraction

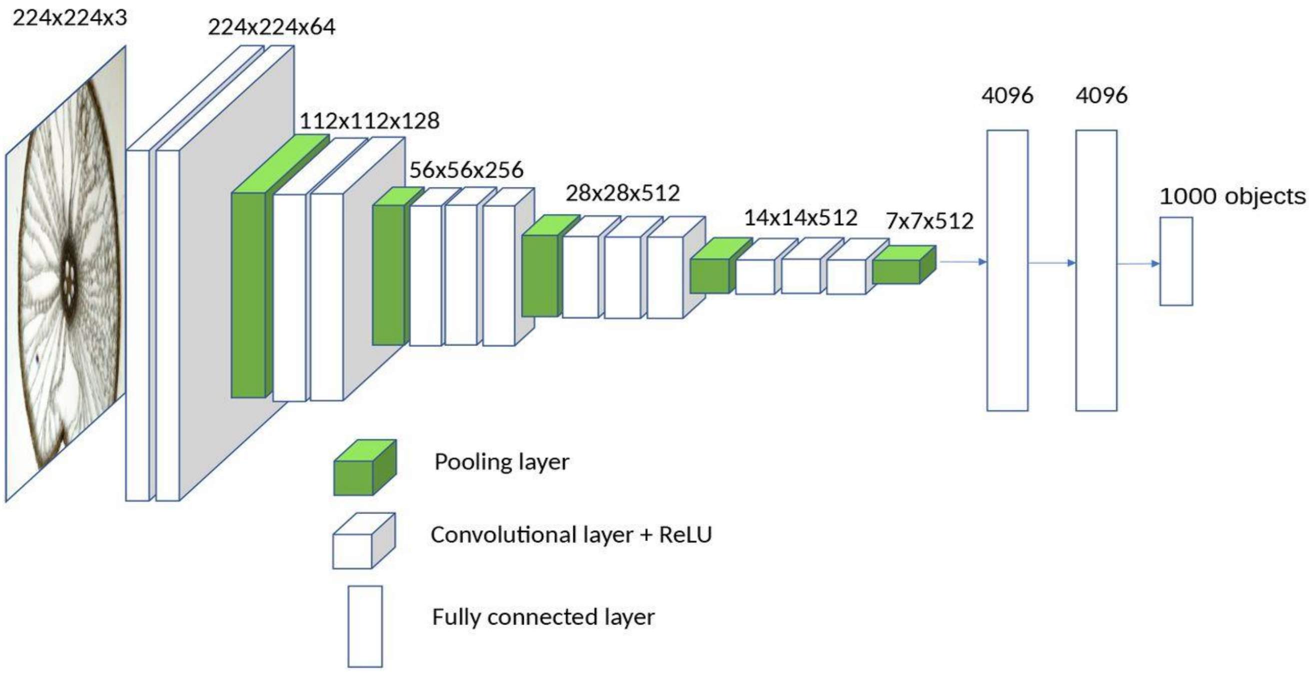
Image feature extraction is the process of identifying and extracting important visual patterns or features from an image, which can be used to represent the image in a compact and meaningful way. The process of feature extraction involves transforming the raw image data into a set of feature vectors that capture the key characteristics of the image. There are various techniques used for feature extraction, including:

1. *Histogram of Oriented Gradients (HOG)*: This technique involves computing a histogram of gradients in different regions of an image to represent the local edge directions.
2. *Scale-Invariant Feature Transform (SIFT)*: This technique involves identifying key points in an image that are invariant to scale, rotation, and illumination changes.
3. *Convolutional Neural Networks (CNN)*: CNNs are a type of deep learning technique that can automatically learn hierarchical features from raw image data.
4. *Local Binary Patterns (LBP)*: This technique involves computing a binary code for each pixel in an image based on the values of its neighboring pixels.

In this project, we will ultilize some state-of-the-art of *Convoluational Neural Networks* to extract the feature of image that we use to build *CBIR* system. Specifically, our team will develop image retrieval systems based on some deep convolutional neural network models such as: *VGG-16, Xception, InceptionV3, InceptionResNetV2, ResNet152V2 and EfficientNetV2L.* We will present the basic knowledge of the methods and apply them to image feature extraction for image retrieval system.

2.1.1. VGG-16

VGG-16 that was introduced in 2014 by a group of researchers at the University of Oxford. It is named after the Visual Geometry Group (VGG) at Oxford. The *VGG-16* model is composed of 16 layers, including 13 convolutional layers and 3 fully connected layers. It can be used to extract the image features in image retrieval system. The process of image feature extraction using *VGG-16* include passing the image through a *CNN* network and taking the output of one or more layers to form feature vectors.



Specifically, to use *VGG-16* for image feature extraction, we do the following:

1. *Image preprocessing*: The image needs to be preprocessed to have a format suitable for the CNN network. This may include converting the image to *RGB* format, converting the image size to match the input dimensions of the *VGG16* (typically *224x224 pixels*), and normalizing pixel values.

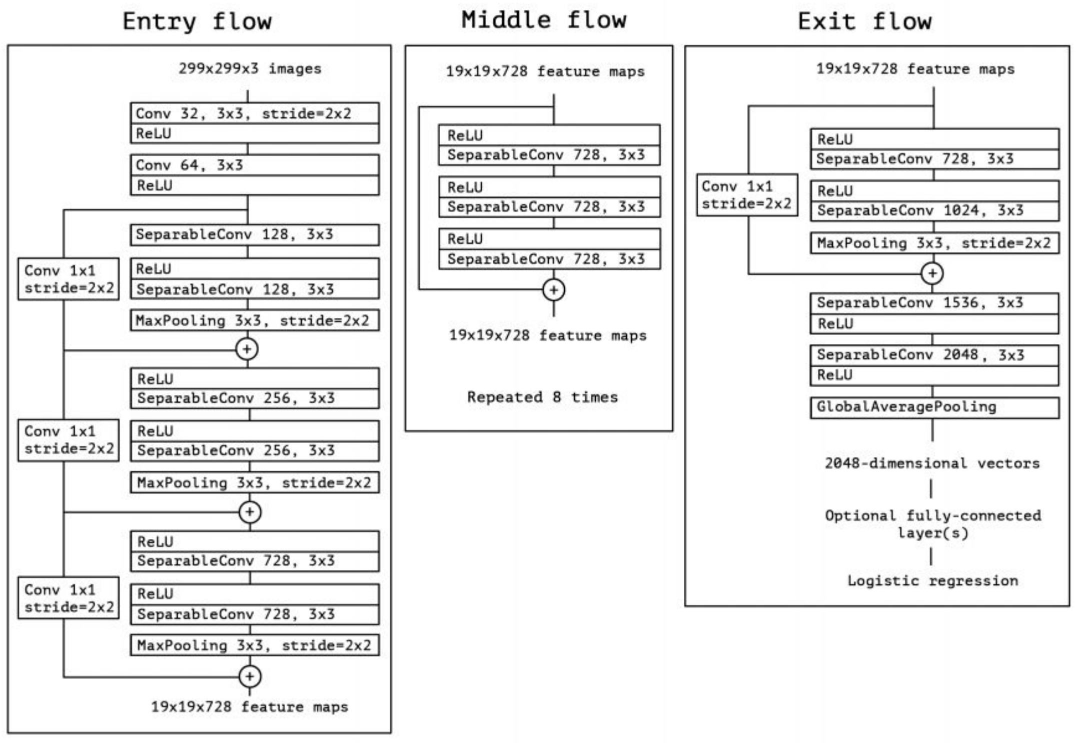
2. *Feature Extraction*: The image is passed through the *VGG16* network and computes the output of one or more layers. These outputs are used to form feature vectors of the image.
3. *Feature storage*: Feature vectors of the images in the dataset are stored for use in image retrieval.
4. *Image query*: The query image is passed through the VGG16 network to calculate the corresponding feature vector. Then, this feature vector is compared with the feature vectors of the images in the data set to find the most similar images.

VGG-16 has been used in many image retrieval systems, with very good results on many test datasets. However, when using VGG16 in an image retrieval system, storing feature vectors of the dataset can consume a lot of memory space, so it should be considered in system design.

In this experiment, we will utilize the weights from the pre-trained model on the "ImageNet" dataset, and we will use the output of the VGG-16 network's first fully-connected layer following the flatten layer as a feature vector. Then we will normalize the feature vectors and save them to use for querying in the following steps.

2.1.2. Xception

Xception is a convolutional neural network (CNN) model introduced by *Google* in 2016, which is one of the CNN models used to extract image features in image retrieval system. Similar to other CNN models, *Xception* can be used to extract image features by passing the image through a CNN network and taking the output of one or more layers to form feature vectors. To use *Xception* to extract image features in an image query system, the steps can be similar to when using *VGG-16*. Nevertheless, the *Xception* network's input is a *299x299 pixels*, we'll need to resize all of the photos in the database to that size before we can utilize it. We will continue to use the pre-trained model weights on the "ImageNet" dataset, and we will get the output results at the *GlobalAveragePooling layer* at *Exit flow* to construct a feature vector of size 2048, similar to feature extraction from the VGG-16 network. Then, we use methods like PCA or LSH to reduce the data dimension and find similar images in the image database.



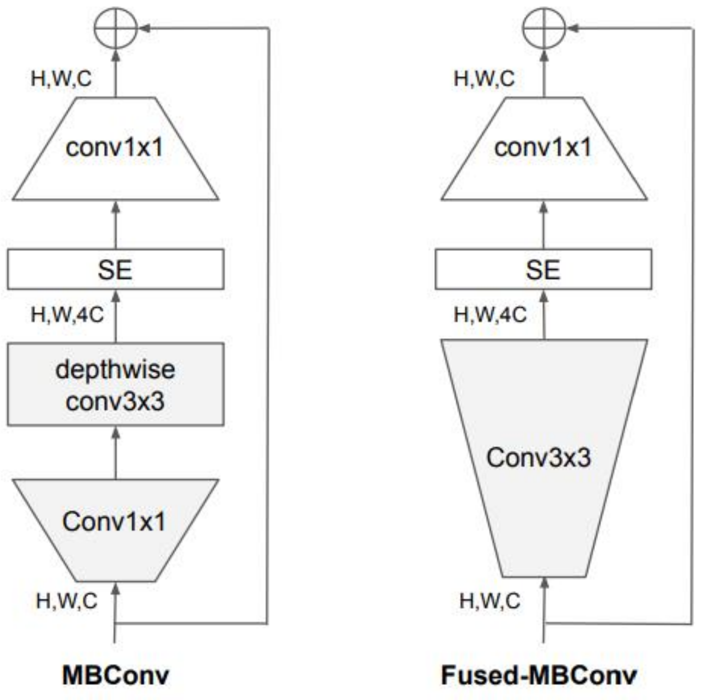
Compared with other *CNN models*, *Xception* can provide higher accuracy in feature extraction. Although *Xception* can be more time consuming to train and requires more computational resources due to the larger number of parameters, in the image retrieval system using Xception for feature extraction can improve the accuracy and performance of the system.

2.1.3. EfficientNetV2L

EfficientNetV2 is the latest of *CNN* models developed by *Google Brain* research team in 2021, it is an upgraded version of *EfficientNetV1* with many improvements and perfections. In particular, *EfficientNetV2L* is a version of *EfficientNetV2*, which is larger in size and suitable for tasks requiring high resolution and good object recognition. *EfficientNetV2L* has a deeper architecture than its predecessor, with a total of 474 million parameters. Its architecture consists of a base network (backbone) and an output network (head).

To extract image features, we can use the underlying network part of *EfficientNetV2L*, namely convolutional layers, with weights trained on the ImageNet dataset. Before using *EfficientNetV2L*, we need to preprocess the image to convert the image size to the input size of the model (*480x480 pixels*). Then, we pass the image through the underlying network of *EfficientNetV2L* and extract the features extracted from the convolutional layers.

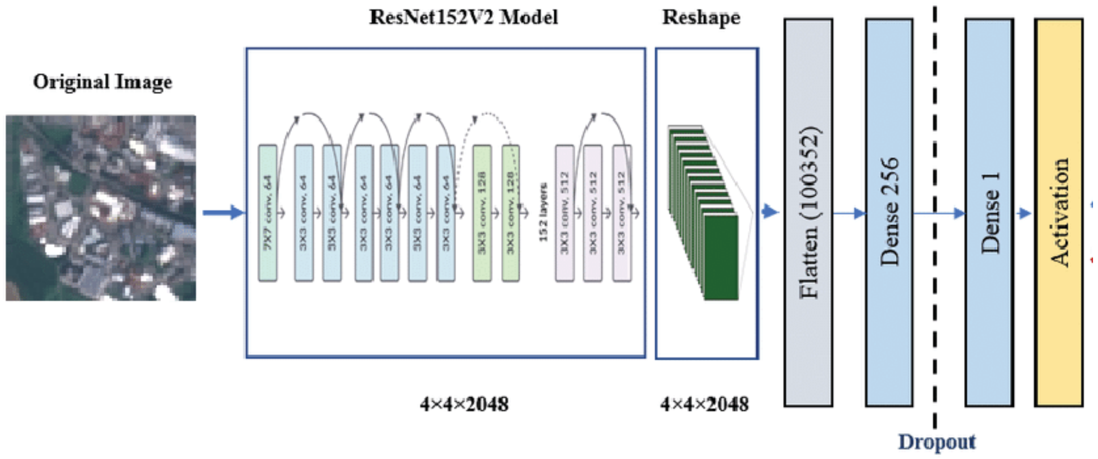
These features are used to compute the feature vector of the image, using a pooling layer, such as global average pooling (GAP). This feature vector can be used to compare images with each other, to find similar images in image retrieval systems.



Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

2.1.4. ResNet152V2

ResNet152V2 is a very popular and widely used deep convolutional neural network (CNN) model in computer vision applications. This is an upgraded version of *ResNet152*, suggested by *Kaiming He et al.* in 2017. This model has 152 network layers and uses the main architecture of *ResNet* with *Bottleneck* blocks.

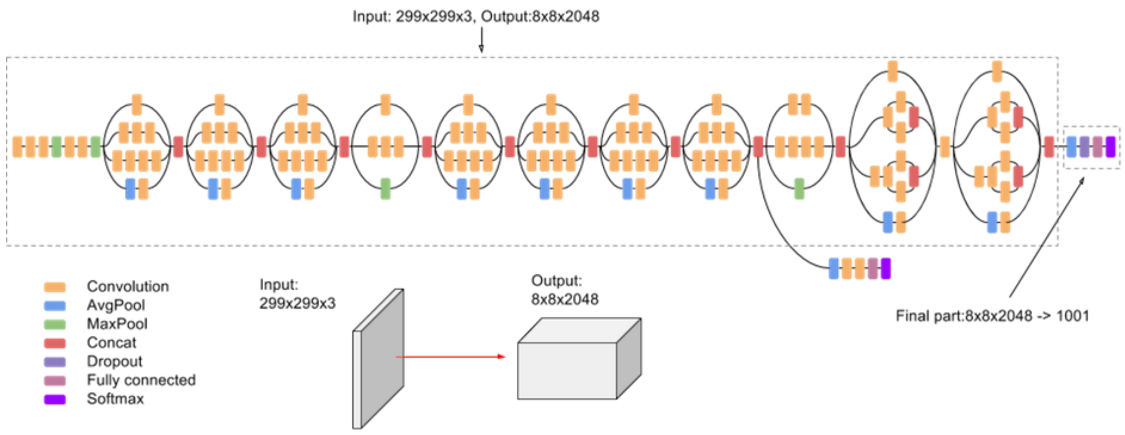


To build an image retrieval system using *ResNet152V2* to extract image features, we resize the images to the same dimensions to give in the model, with input image *224x224 pixels*. Then we normalize the pixel values of the image to the interval [0, 1] to include in the model. After that, we run each image through the model and collecting the output from an intermediate layer in the model named *"avg_pool"*. This is an averaging pooling layer (i.e. input size reduction class) that reduces the output size of *ResNet152V2* from *(7, 7, 2048)* to *(1, 1, 2048)*, the average representation of the values per channel of the last feature map. In this case, each image is converted a feature vector with size *2048*.

2.1.5. InceptionV3

InceptionV3 is a deep neural network architecture which was built by Google and is an improved version of the previous *Inception* architecture, with better performance and higher accuracy. The *InceptionV3* architecture has 48 layers and is built on the idea of using *Inception* modules, each of which consists of convolution and pooling layers of different sizes, stacked on top of each other for image feature extraction. This architecture also uses techniques such as batch normalization, regularization, and skip connections to reduce vanishing gradients and improve network performance.

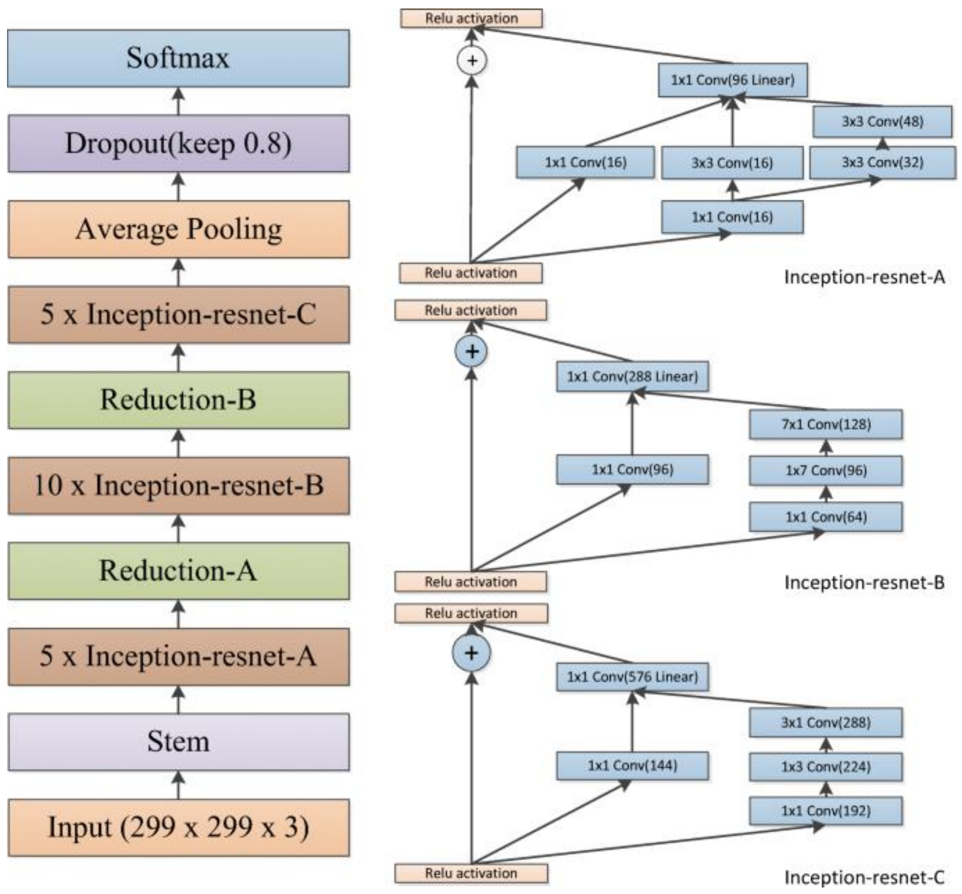
InceptionV3 was also trained on the *ImageNet* dataset with over 1.2 million images and 1000 classes. The default input size of *InceptionV3* is *299x299 pixels* and the output of the architecture is a 1-dimensional feature vector with *2048* elements.



2.1.6. InceptionResNetV2

InceptionResNetV2 is a deep neural network architecture built by *Google* and combines ideas from the *Inception* and *ResNet* architectures. It has 572 layers and uses *Inception* modules with skip connections and *ResNet* modules to improve accuracy and reduce vanishing gradients. This architecture also uses techniques such as batch normalization and regularization to reduce overfitting. *InceptionResNetV2* is trained on the *ImageNet* dataset and the output of the architecture is a 1-dimensional feature vector with size *1536* which used to evaluate similarity between images and perform image retrieval tasks.

To extract image features using *InceptionResNetV2,* we use the *Keras* library to load the *InceptionResNetV2* model and specify the final layer as *GlobalAveragePooling2D* to get a 1-dimensional feature vector with 1536 elements for each image. Then, we can pass the image to be extracted into the model and use the predict method to get the corresponding feature vector.



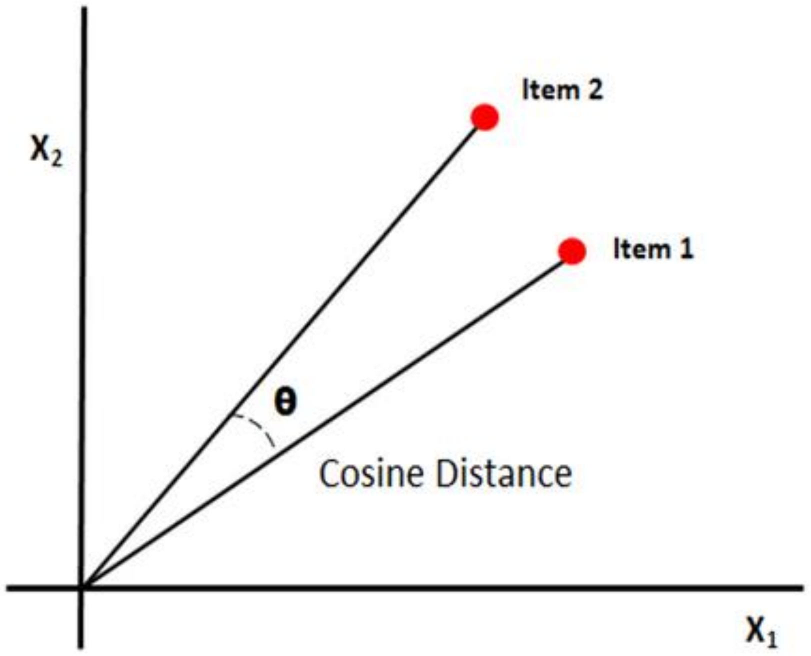
2.2. Similarity calculation

Once we have a database of feature vectors, we need to determine the distance measure (or similarity measure) so that we can compare the feature vectors of the images in the corpus with the feature vectors of the query image, to find the images that are most similar to the query image.

Our image search system uses two common distance measures, Cosine Distance and Euclidean Distance. Feature vector of the query image after extraction will be compared with all available feature vectors in the database, the results obtained will be ranked from smallest to largest because the two images are more similar. when the distance between two feature vectors is smaller.

2.2.1. Cosine distance

Cosine similarity is the measure of similarity between two vectors, calculated as the *cosine* of the angle between those two vectors. Two vectors are more similar if the similarity approaches 1 (the angle between the two vectors approaches 0). Therefore, we define the *Cosine distance* as 1 minus the *cosine* of the angle between those two vectors.



With two feature vectors u and v, the cosine distance between them is calculated by the formula:

$$Similarity = 1 - \cos(\theta) = \frac{u.v}{\|u\|\|v\|}$$

However, Cosine similarity only calculates the angle between two vectors and in the direction, not using magnitude, so it will cause information loss leading to reduced performance.

2.2.2. Euclidean distance

Euclidean Distance is the most common and easy to understand method in practice. Used to calculate the length distance of a line segment joining two points, commonly referred to as the L-2 distance.

The formula calculating Euclidean Distance:

$$D(u, v) = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}$$

Euclidean Distance is especially effective with low-dimensional data because it is easy to understand, easy to implement, and gives quite good results. However, *Euclidean Distance* is affected by the unit of feature and the number of dimensions of the vector, so when applied in practice with large and multidimensional datasets, *Euclidean Distance* becomes inefficient.

2.3. System evaluation

As introduced in the previous section, the dataset has 55 pre-assigned ground truth query images containing images that are considered relevant to the query, divided into 4 groups: Good, OK, Junk and Bad based on the level of view clarity of the object. In this project, we will combine two files "good" and "ok" to evaluate the system and ignore “bad” since it is unaffected; each image returned by the system in both result files is counted as "relevant," and if none, it's counted as "irrelevant." On this basis, we will assess the system's performance using two datasets.

We use the *Mean Average Precision* (*mAP*) measure to evaluate the system as well as compare tested methods. It is a method of evaluating the performance of an information query system. In the process of querying information, the system will search for documents related to the query keyword and sort them by relevance.

mAP is used to measure the quality of an information retrieval system by calculating the average of the precision value at different recall thresholds. Precision is the ratio of the number of related documents returned to the total number of documents returned, and recall is the ratio of the number of related documents returned to the total number of related documents.

Mean average precision is calculated as the average of the average precision of the queries. *Average precision* is the average of precision at the positions the return is said to be relevant.

$$AP = \frac{\sum_{k=1}^{n} P(k) * rel(k)}{number\ of\ relevant\ documents}$$

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

Beside the *non-interpolated mAP*, we carry out extra *interpolated mAP* which calculate based on *11-point interpolated average precision*. For each information need, the interpolated precision is measured at the 11 recall levels of 0.0, 0.1, 0.2, …, 1.0. For the precision-recall curve in Figure 3. For each recall level, we then calculate the arithmetic mean of the interpolated precision at that recall level for each information need in the test collection. A composite precisionrecall curve showing 11 points can then be graphed. The *interpolated precision* at a certain recall level *r* is defined as the highest precision found for any recall level $r' \geq r$:

$$p_{inter\,p}(r) = max_{r' \geq r}\ p(r')$$

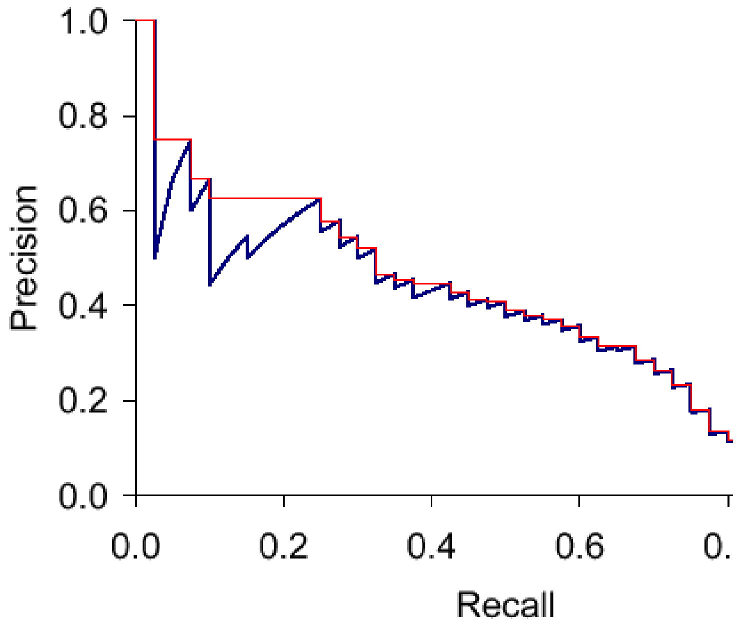


Figure 3 Precision/Recall graph.

2.4. Results, analysis and inferences

After carrying out some experiments, we receive the evaluation results that were calculated based on a set of 55 queries over which an object retrieval system provided being as follows:

- <u>Regarding to *The Oxford Building Dataset*:</u>

Method	**Non-interpolated MAP**	**Interpolated MAP**	**Time (s)**
VGG-16	0.54	0.24	40.77
EfficientNetV2L	**0.71**	**0.37**	**110.96**
Xception	**0.68**	**0.34**	**39.94**
ResNet152V2	0.60	0.26	46.51

InceptionV3	0.62	0.32	36.51
InceptionResNetV2	0.60	0.27	48.57

Detailed results with 55 queries: The Oxford Building Evaluation

- Regarding to *The Paris Bulding Dataset*:

Method	**Non-interpolated MAP**	**Interpolated MAP**	**Time (s)**
VGG-16	0.78	0.43	45.75
EfficientNetV2L	**0.85**	**0.53**	**118.79**
Xception	**0.86**	**0.54**	**44.44**
ResNet152V2	0.84	0.51	57.52
InceptionV3	0.86	0.54	42.71
InceptionResNetV2	0.81	0.47	57.91

Detailed results with 55 queries: The Paris Dataset Evaluation

Through the results obtained from the two datasets above, we have some comments:

- Considering *MAP* (*non-interpolation and interpolation*), *Xception* and *EfficientNetV2L* are two deep neural network models that bring higher results than other methods.

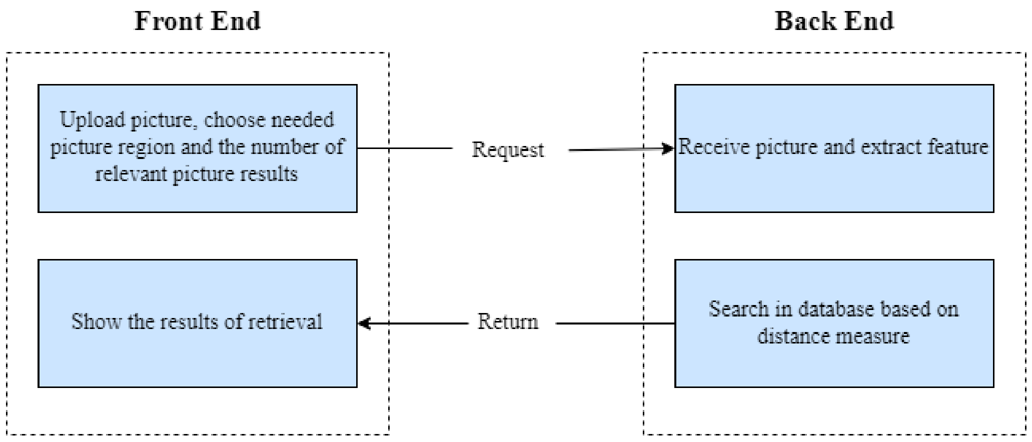
- However, for a retrieval system, the time to search for results is also an important factor in the evaluation, so we find *Xception* somewhat superior to *EfficientNetV2L*.

From the comments, we have concluded that *Xception* will be deploy into the website as image retrieval system with interface and some retrieval functions.

2.5. Demo

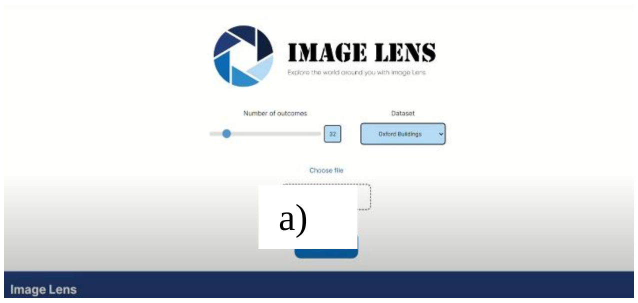
We carried out and recorded our demo as video, following this link to know more details: Demo

We implement the image retrieval system on the website, with an interface to help users easily query and view the returned results. The system is implemented based on the following diagram:

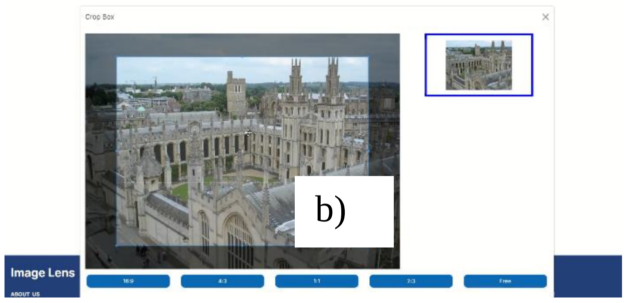

Front End
Back End
Upload picture, choose needed picture region and the number of relevant picture results
Request
Receive picture and extract feature
Show the results of retrieval
Return
Search in database based on distance measure

2.5.1. Front-end

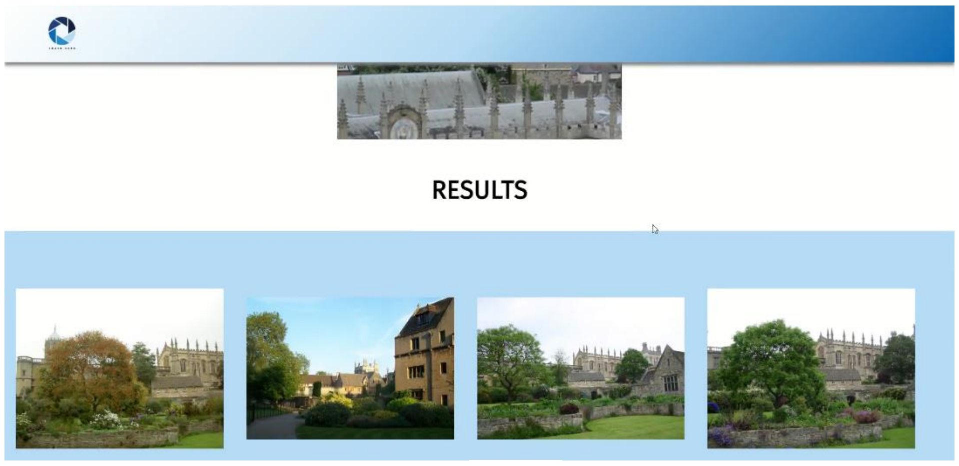
The website is built using *HTML, CSS and JavaScript*. The image is uploaded in the front-end, which has a feature that allows the user to select the area to search and the number of relevant image results, then transmit the required image area to the server, the server will perform the search and returns the number of relevant images follow requirement. These are most similar to the query image so that the website can be displayed to users.



a)



b)



c)

Figure 4 Front-End Image Retrieval System

a) Home page b) Select image area to search
c) The results of image retrieval system

2.5.2. Back-end

The system uses Python language to operate the server and Flask framework to send and receive requests between front-end and back-end. The server receives a request that the image has been processed, the image is extracted using the deep neural network (Xception) and compares the similarity with the features of the previously extracted dataset using the Cosine distance. The results of retrieval which equal to the number of require are sent to the front-end to display on the screen.

III. Conclusion

Image retrieval is a problem that has been research and has significant applications in today's life. Organizing and searching images quickly will assist people in accessing and capturing information better. Through this project, our team has studied, researched and developing deep learning network models to build image retrieval system. Besides the experiment and evaluation on the required dataset (*The Oxford Building Dataset*), we also built models on *The Paris Building Dataset* and obtained some positive results. Not only do we stop at building and evaluating methods, we also deploy them into a website with interface and image retrieval functions. In the future, we will develop more systems to increase the accuracy and speed of processing by studying and using more modern deep learning networks to extract high-level features, combined with the use of more sophisticated measures to compare feature vectors (such as *triplet loss*), and additional use of search techniques on large image databases.

REFERENCES

[1] "Image Feature Extraction: Traditional and Deep Learning Techniques," 9 September 2020. [Online]. Available: https://towardsdatascience.com/image-feature-extraction-traditional-and-deep-learning-techniques-ccc059195d04.

[2] "Thử làm Google Image Search (Content based Image Retrieval) - Mì AI," 2 October 2021. [Online]. Available: https://miai.vn/2021/10/02/thu-lam-google-image-search-content-based-image-retrieval-mi-ai/.

[3] A. Rosebrock, "Keras: Feature extraction on large datasets with Deep Learning," 27 May 2019. [Online]. Available: https://www.pyimagesearch.com/2019/05/27/keras-feature-extraction-on-large-datasets-with-deep-learning/.

[4] "Keras Applications," [Online]. Available: https://keras.io/api/applications/#usage-examples-for-image-classification-models.

[5] Wan J, Wang D, Hoi SCH, Wu P, Zhu J, Zhang Y, Li J. Deep learning for content-based image retrieval: a comprehensive study. In: ACM international conference on multimedia 2014. [Paper]

[6] Huang W, Qiang W. Image retrieval algorithm based on convolutional neural network. In: Selected paper from Common Sense Media Awards 2017. [Paper]

[7] Hanan A. Al-Jubouri, Sawsan M. Mahmmod. "A comparative analysis of automatic deep neural networks for image retrieval". [Paper]

[8] Nikhil V Shirahatti, Kobus Barnard. "Evaluating Image Retrieval". [Paper]

[9] Mayank R. Kapadia, Dr. Chirag N. Paunwala. "Content based medical image retrieval system for accurate disease diagnoses using modified multi feature fused Xception model". [Paper]