VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

Project::

INFORMATION RETRIEVAL SYSTEM

INSTRUCTOR: Dr. NGO DUC THANH

STUDENTS: BUI QUANG PHU - 20520273

DO THI THU TRANG - 20520816

NGUYEN MINH THUAN - 20520795

NGUYEN DANG BAO NGOC - 20521663

Ho Chi Minh, February 20th, 2023

TABLE CONTENT

I. Overview

1. Introduction

Information retrieval is a key task in computer science that involves searching and retrieving relevant information from large datasets. Information Retrieval has some common applications are search engine, recommender system, question answering systems, text classification and clustering, … An image search engine is a useful tool for finding and retrieving specific images from a large image dataset. Image retrieval is the task of searching for images that are visually like a given query image.

In this project, we aim to build an image search engine based on the image retrieval technique. We will use three pre-trained deep learning models - Xception, VGG16, and EfficientNetV2 - to extract features from images, which will be used to represent the images in a high-dimensional space.

We will evaluate the performance of our image search engine using the Oxford Building dataset, which is a commonly used dataset for image retrieval tasks. We will use mean average precision (MAP) as the evaluation metric, which measures the average precision of the retrieved images across all queries.

Our image search engine will be able to efficiently and accurately retrieve images that are visually like a given query image, which can be useful in a wide range of applications such as image classification, image recognition, and recommendation systems. By using state-of-the-art deep learning models and a well-established evaluation metric, we aim to build an image search engine that achieves high performance and is reliable for real-world use cases.

2. Dataset

The Oxford Building dataset is a collection of images of buildings in the Oxford area. It was created by the Department of Engineering Science at the University of Oxford in the UK for the purpose of research in computer vision and image processing. The dataset consists of 5062 high-resolution images of 11 different buildings, captured from different viewpoints and under varying lighting conditions, each represented by 5 possible queries. This gives a set of 55 queries over which an object retrieval system can be evaluated.

The buildings in the dataset are: All Souls College, Ashmolean Museum, Balliol College, Corpus Christi College, Hertford College, Jesus College, Keble College, Magdalen College, New College, Oriel College, and University Church of St Mary the Virgin.

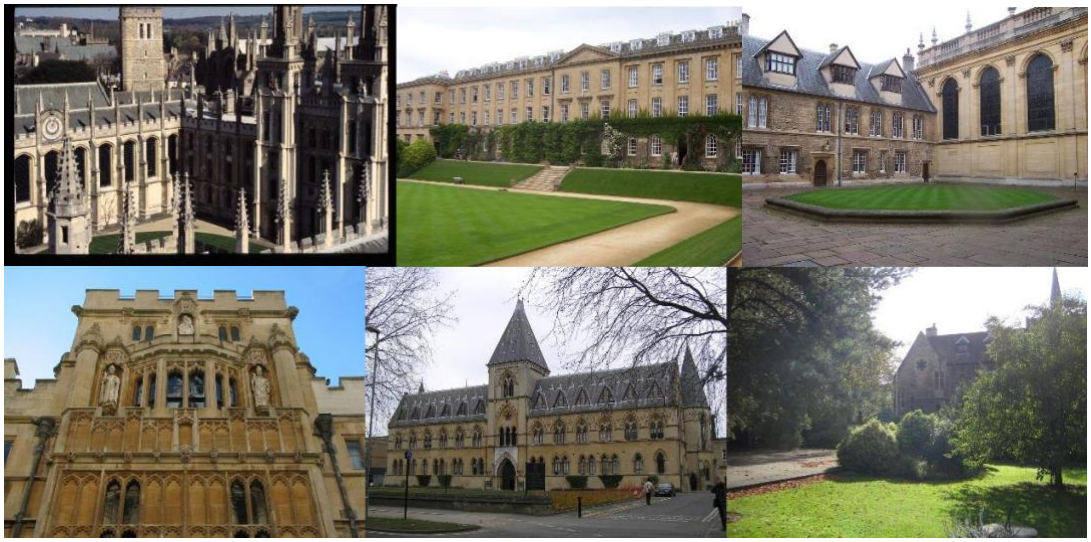

Figure 1: Images of buildings in dataset

Each image in the dataset is labelled with the name of the building it depicts, and the viewpoint from which it was captured. In addition, the dataset includes ground truth information for the location and orientation of the camera relative to the building.

For each image and landmark in dataset, one of four possible labels was generated:

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

One of the challenges of working with the Oxford Building dataset is the large amount of variation in lighting conditions and viewpoints, which can make it difficult to develop robust algorithms that can handle these variations.

II. System approach

1. Workflow

To perform image retrieval, we extract the deep features from the pre-trained models, which are the output of the last convolutional layer before the fully connected layers. We use the Cosine distance to measure the similarity between the deep features of the query image and the database images. Specifically, given a query image, we compute the distances between its deep features and the deep features of all the database images. We then sort the database images based on their distances to the query image and return the top-k images as the retrieval results.

In more detail, the workflow can be summarized as follows:

1. Load the pre-trained deep learning model (VGG16, Xception, or EfficientNetV2).
2. Remove the fully connected layers from the model and get the output of the last convolutional layer.
3. Extract the deep features of all the images in the database using the pre-trained model.
4. Given a query image, extract its deep features using the pre-trained model.
5. Compute the Cosine distances between the query image's deep features and the deep features of all the images in the database.
6. Sort the database images based on their distances to the query image.
7. Return the top-k images as the retrieval results.

2. Models

For this image search engine, we will be using the image retrieval technique, which involves finding images that are like a query image based on their visual content. Specifically, we will use the Xception, VGG16, and EfficientNetV2 deep learning models to extract features from the images, and then use those features to compute the similarity between the query image and the images in our database. These models are pre-trained on the ImageNet dataset, which contains millions of images and thousands of object categories.

a. VGG16

VGG16 is a convolutional neural network (CNN) architecture that was introduced by the Visual Geometry Group at the University of Oxford in 2014. It was designed for image recognition and classification tasks and has achieved state-of- the-art performance on several benchmark datasets.

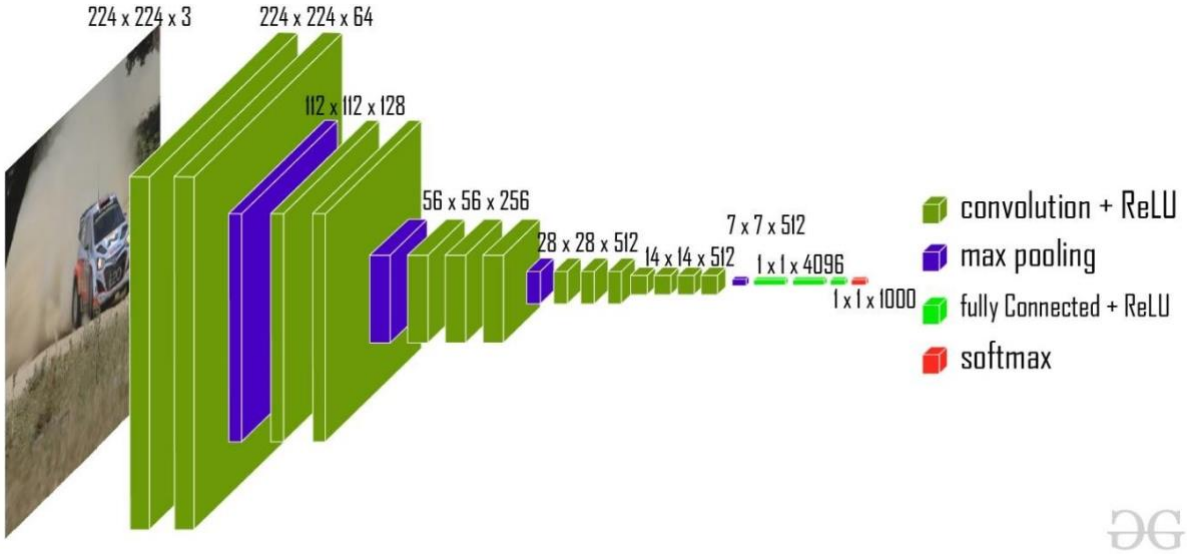

Figure 2: VGG16 architecture

The network consists of 16 convolutional layers followed by 3 fully connected layers. Each convolutional layer uses a 3x3 kernel with stride 1 and is followed by a rectified linear unit (ReLU) activation function and a 2x2 max pooling layer with stride 2. The first two fully connected layers have 4,096 neurons each, and the final fully connected layer has 1,000 neurons for classification.

VGG16 is a deep and complex network with over 138 million parameters and requires a large amount of training data and computational resources to achieve good performance. However, it has proven to be a highly effective architecture for image classification, achieving top-5 accuracy of over 90% on the ImageNet dataset, which contains over 1.2 million images in 1,000 categories.

In the context of image search engines, VGG16 can be used as a feature extraction method to map images to high-dimensional vectors, which can then be compared and ranked to find similar images. The VGG16 features can be extracted from the last fully connected layer or from an intermediate convolutional layer.

b. Xception

Xception is a deep learning-based method for image retrieval that involves using convolutional neural networks (CNNs) to extract features from images, followed by a similarity search to find images that are most like a query image. This method was proposed by researchers at Google in 2016 and has been shown to be highly effective in a number of image retrieval tasks.

To use Xception for building an image search engine, the first step is to train a CNN on a large dataset of images to extract features from each image. The Xception architecture is similar to other CNNs such as VGG16 and EfficientNetV2, but it uses a different approach to convolution called depthwise separable convolutions. This allows it to achieve high accuracy with fewer parameters than other CNNs.

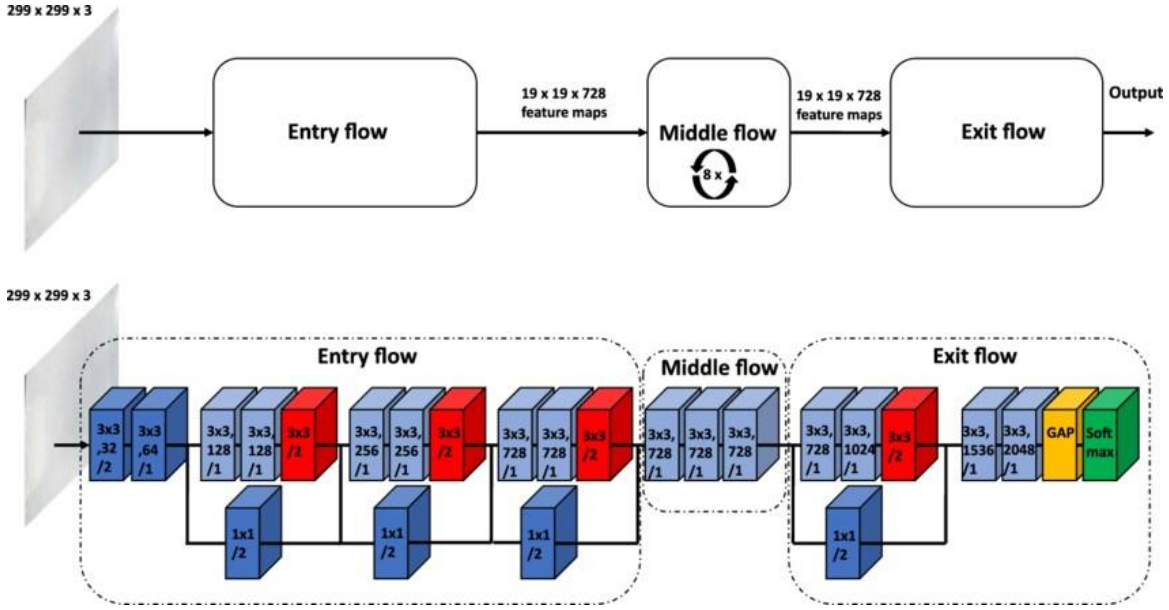

Figure 3: Xception architecture

Depthwise separable convolutions are a type of convolution that factorizes a standard convolution into two parts: a depthwise convolution and a pointwise convolution.

The depthwise convolution applies a single filter to each input channel, resulting in a set of intermediate feature maps. The pointwise convolution then applies a 1x1 convolution to these feature maps to combine the information across channels.

The benefit of using depthwise separable convolutions is that they are significantly more efficient than standard convolutions, requiring fewer parameters and less computation. This can lead to faster training times and better performance on devices with limited computational resources.

In the context of image search engines based on image retrieval, the use of depthwise separable convolutions in Xception can allow for faster and more efficient feature extraction from images. This, in turn, can lead to faster and more accurate image search results.

c. EfficientNetV2

EfficientNetV2 is a state-of-the-art deep neural network architecture for image classification and other computer vision tasks. It was first proposed in 2020 as an improved version of the original EfficientNet architecture.

EfficientNetV2 is a family of convolutional neural network architectures that was proposed as an extension to the EfficientNet architecture. The original EfficientNet was designed using a neural architecture search (NAS) to balance model complexity and accuracy, resulting in a highly efficient and effective model. EfficientNetV2 builds upon the original EfficientNet architecture by incorporating several new techniques and design principles to improve its performance.

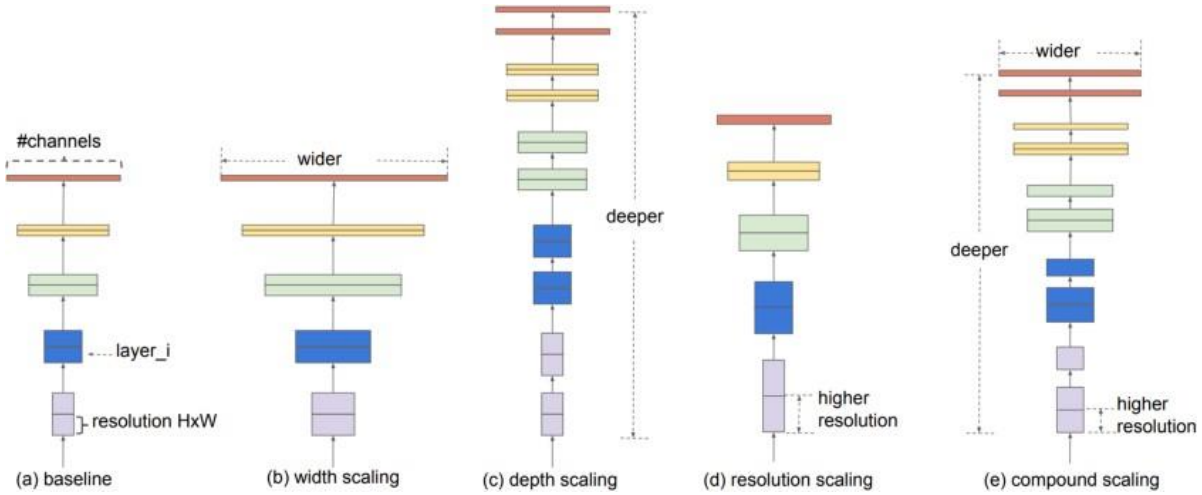

Figure 4: Scaling the depth, width, and image resolution to create different variations of the EfficientNet model.

EfficientNetV2 uses a combination of three key techniques to improve its performance: (1) Compound scaling, (2) Advanced regularization, and (3) Stochastic Depth.

Compound scaling involves scaling the network's depth, width, and resolution simultaneously, rather than separately as in previous models. This allows for a more efficient use of resources, resulting in higher accuracy with fewer parameters.

Advanced regularization techniques are used to prevent overfitting, including a novel technique called DropConnect, which randomly drops connections within the network during training, and weight decay.

Stochastic Depth is a technique that randomly skips a block of layers during training, which helps to avoid overfitting and improves performance.

EfficientNetV2 uses a highly efficient architecture, consisting of a series of convolutional layers with depthwise separable convolutions, which significantly reduces the number of parameters and computational cost. This architecture allows for faster training and inference times, making it ideal for large-scale image retrieval applications.

3. Step by step

a. Image preprocessing

Image in dataset will be processing through the flow:

- Resize: with model VGG16, image size is (244, 244); with model Xception image size is (299, 299) and model EfficientNetV2 has image size is (480, 480)
- Convert RGB: although, input data is RGB image, all the input data is not secured that is RGB Image. So, we need to convert image into RGB Image.
- Change image to 3-dimension array: that image after be preprocessed will be change to array for extracting feature and stored.
- Add more dimension: Our model using TensorFlow, so we need to expand array dimension. The new dimension is the number of the image.

b. Feature extraction

These models have been pre-trained on large datasets and are known to be effective at extracting high-level features from images.

The output of each model will be a high-dimensional feature vector that represents the visual content of the image. We will then use these feature vectors to compute the similarity between images.

c. Feature storage

Feature of that picture are saved into NumPy array file. The result of each model will be stored into 1 folder, name of that folder is model name. The feature files are be named with structure [image name].npy

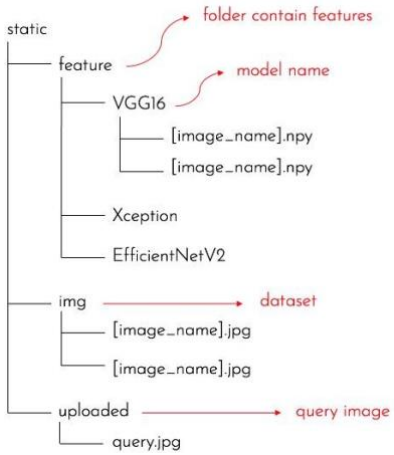

Figure 5: Tree structure of data storage

d. Image query

Image area needed to find is cut by library cropper.js. Then, that image is saved and processed through flow: image preprocessing, feature extraction and then be computed similar to each image feature in dataset. The result is k images that have highest score.

4. Similarity calculation

Given a query image, we will compute its feature vector using each of the three models, and then compute the cosine similarity between the query feature vector and the feature vectors of all the images in our database. We will then rank the images in our database based on their similarity to the query image.

To compute the similarity between the query image and the images in our database, we will use the cosine similarity measure. The cosine similarity measures the cosine of the angle between two vectors and is a common way to compare the similarity between high-dimensional feature vectors.

$$\text{cosine similarity} = S_C(A, B) = \cos(\theta) = \frac{A \cdot B}{|A||B|}$$

A high cosine similarity score indicates that the two images are similar, while a low score indicates that they are dissimilar. Typically, a similarity threshold is used to determine when two images are similar enough to be considered a match.

5. Evaluation

We evaluate the performance of the three deep learning models using the Mean Average Precision (MAP) metric.

Mean Average Precision (MAP) is a popular evaluation metric used to evaluate the performance of information retrieval systems, including image search engines. MAP is calculated by computing the average of Average Precision (AP) scores over a set of queries.

AP score for a single query is calculated by ranking the retrieved images in order of relevance to the query, then computing the precision and recall for each retrieved image and taking the area under the precision-recall curve. The precision is the fraction of retrieved images that are relevant, while the recall is the fraction of relevant images that are retrieved.

MAP is particularly useful when evaluating information retrieval systems on datasets where there are multiple relevant items for each query. It takes into account the order of the retrieved items, and therefore provides a more accurate assessment of the system's ability to retrieve relevant items.

MAP Interpolated is a popular evaluation metric for image search engines that measures the effectiveness of the system by considering the precision at different recall levels. It is an improved version of the Mean Average Precision (MAP) metric that provides a better understanding of the system's performance.

	Non–Interpolated MAP	*Interpolated MAP*	*Evaluating time*
VGG16	0.54	0.23	386.19s
Xception	0.68	0.34	323.16s
EfficientNetV2L	0.43	0.13	2215.27s

The results showed that the Xception method has the best performance: the highest MAP Interpolated score of 0.34, the highest MAP score of 0.68 and the fastest with 323.16 seconds.

Overall, the results suggest that the Xception method is the most effective for image retrieval in terms of accuracy and speed. However, further research may be required to determine the most effective method for different types of images and queries.

III. Results

We built a webserver through localhost. You can send your query image to the server via a Flask web-interface. The server finds similar images to the query by a simple linear scan.

1. Front-end

We use HTML, JavaScript, CSS to build User Interfaces. This UI allow user to upload image and choose image area for retrieval.

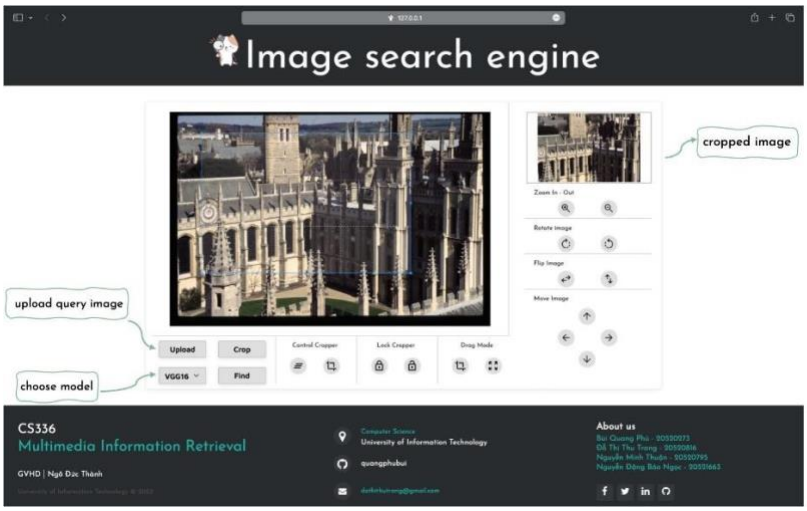

Figure 6: User interface

2. Back-end

Get the image that cropped in frontend from folder has name “uploaded”. Then, extract it into feature and compare with each image in dataset. After that, show k images that more similar to query image. Finally, push that result to user interface.

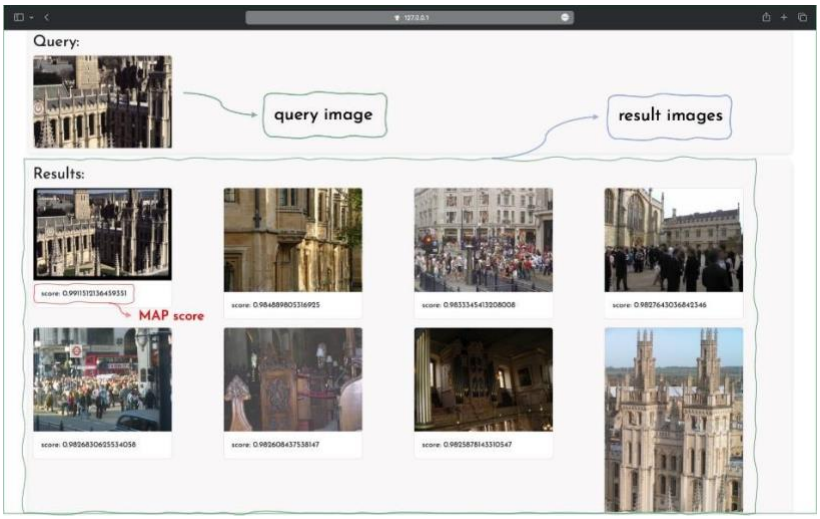

Figure 7: Result of image retrieval

3. Demo

Experiment result will be demo through link: [Demo Information Retrieval Image]

IV. Conclusion

In conclusion, building an image search engine is a complex process that involves several steps, including data collection and preprocessing, model selection, and evaluation. In this project, we used the Oxford building dataset to evaluate the performance of three popular image retrieval models: Xception, VGG16, and EfficientNetV2.

Our evaluation focused on the Mean Average Precision (MAP) score and Mean Average Precision Interpolated (MAPI) which are common metrics for measuring the performance of image retrieval systems. We found that all three models achieved high MAP scores, with Xception outperforming the other two models.

However, it's worth noting that the performance of the search engine also depends on the quality and diversity of the dataset used for evaluation. Therefore, further research could involve testing the models on other datasets to ensure their generalizability.

In summary, our project demonstrated the potential of using deep learning models for image retrieval in building an image search engine. With further research and development, this technology could have practical applications in a range of fields, including e-commerce, art and design, and security.

V. References

[1] Simple Image Search Engine, matsui528, https://github.com/matsui528/sis

[2] Crop Image in Vanilla JavaScript Full Project, Hassnain Amjad, https://www.slidesmaker.me/blog/crop-image-in-vanilla-javascript-full-project

[3] EfficientNet V2 models for Keras, edumucelli, https://github.com/keras-team/keras/blob/v2.11.0/keras/applications/efficientnet_v2.py#L1294-L1321

[4] Instantiates the VGG16 model, haifeng-jin, https://github.com/keras-team/keras/blob/v2.11.0/keras/applications/vgg16.py#L48-L252

[5] Instantiates the Xception architecture, haifeng-jin, https://github.com/keras-team/keras/blob/v2.11.0/keras/applications/xception.py#L50-L360

[6] The Oxford Buildings Dataset, James Philbin, Relja Arandjelović and Andrew Zisserman, https://www.robots.ox.ac.uk/~vgg/data/oxbuildings/

[7] The Oxford Buildings Dataset Ground truth file, James Philbin, Relja Arandjelović and Andrew Zisserman, https://www.robots.ox.ac.uk/~vgg/data/oxbuildings/gt_files_170407.tgz