Content-based Image Retrieval

Haoran Wang, hw1393@nyu.edu
Yuanbin Wang, yw2303@nyu.edu

This is a project report for the graduate capstone course of Search Engine Architecture at NYU. In this project, we implemented a basic image search engine based on mere image content. In other words, we use an image to search other similar images. We don’t take into consideration the text information related to images, but just the image content. As the past few years have seen amazingly successful applications of deep learning to images, we employ convolutional neural networks as our tool to extract content (features) from images. We chose this topic for our project specifically because information (image) retrieval is highly relevant to this course and also because we would love to explore the use of deep learning for content-based image retrieval. Our results show that our image search engine can correctly retrieve similar images given an query image, which further proves the use of deep learning as image feature extractor in this case. Next, we will briefly describe the details of the project in the following sections.

Additional Key Words and Phrases: Image retrieval, deep learning, image content, search engine, information retrieval, search engine architecture, capstone project

ACM Reference Format:
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Information retrieval, especially text retrieval, has become one of the most influential and beneficial technologies in the past decades. Together with the development of Internet, it helps people all over the world obtain all kinds of information easily almost without any cost. This kind of information democratization has become the powerful fuel for the economical, cultural, and technological development of human beings. At the same time, images also have become a significant type of information on the Internet because of the rapid development of Internet, digital camera, and multimedia social network. It’s now a trend that people are more and more immersed in the image-based social network applications, such as Instagram, Snapchat, Flickr, etc. Some world-leading technology giants such as Google1 and Baidu2 have devoted their efforts to this area.

Motivated by this phenomenon and the huge success of text search in the past decades, we built an content-based image search engine on our own using information retrieval knowledge learnt in the class. Not only that, we also integrated deep learning as a tool to represent an image into our project, as it’s been so successful in different kinds of tasks of computer vision recently such as image classification, image detection, image segmentation, etc.

Specifically, we used a pre-trained deep learning model to extract features from the images on which it’s directly trained on. The training task is classification, but we believe that the features/representation codes we get from the model can also apply to other tasks as well, such as image retrieval. We used those features as a representation for images. Once we get a new query image, we get its feature using the pre-trained model, and we compare it to all image we have based on their similarity scores between the features of each other. Finally we retrieve top K images and display them

---

1 https://images.google.com/
2 https://image.baidu.com/
on the web page. We will talk about the details of datasets, the deep learning model, frameworks, and search engine architecture below.

2. DATASET AND MODEL

2.1. ImageNet

ImageNet\(^3\) is an image database that contains more than 1 million images that are distributed to 1000 categories. ImageNet Large Scale Visual Recognition Challenge, which is usually called ILSVRC\(^4\), has become a place where computer vision researchers around the world compete against each other in object detection and classification tasks. We chose it as our image database because

1. It’s large and diversified enough for the image retrieval task in our project
2. On the other hand, it’s not too large to fit into the disk space on a desktop, due to the restriction of computation resources
3. Last but not least, there are many pre-trained models trained on ImageNet, so that we can just pick one and use it without having to train the model starting from scratch or fine-tuning the model.

In our project, we only use training data of ImageNet.

2.2. Residual Network

Residual Network \([\text{ResNet}]\) is the current state-of-the-art deep learning model on ILSVRC classification task. Generally, it’s designed so that it can be optimized more easily and many more layers can be added to achieve better performance. It’s been a revolutionary innovation and many researchers have proposed other similar architectures based on it to achieve better performance since then. A typical basic building block of a residual network is shown in Figure 1. A residual network contains multiple such building blocks by combining them in several ways together. The authors of the model claimed that they achieved 3.57\% error rate on ImageNet test set.

Specifically, we chose ResNet-50\(^5\) as our model based on the following reasons

1. The performance of the model is one of the very best among many others
2. The model is not too large to compute feature extraction for a query image in our real-time application
3. Its pre-trained model on ImageNet is readily available in Keras, which is a convenient wrapper for Tensorflow

Although the model is designed for image classification, the internal representation of images it has learned can be applied to other tasks as well. However, in our image

\(^3\)http://image-net.org/
\(^4\)http://image-net.org/challenges/LSVRC/
\(^5\)http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006
retrieval task, it does show a small bias towards retrieving the images belonging to
the same category as the query image when using original features from the model.
This bias can be mitigated when we reduce the dimensions of features by PCA. We will
discuss this phenomenon in the DISCUSSION section.

3. FRAMEWORKS AND ARCHITECTURE

3.1. Frameworks

3.1.1. Python. We use Python as our main language to build the whole framework

3.1.2. Tornado. We use Tornado as a web framework to build the search engine. It
supports asynchronous networking, which can scale to tens of thousands of open con-
nections.

3.1.3. HTML, CSS, Javascript. We use them to build front-end web interface to users.

3.1.4. Keras and Tensorflow. We use Keras as a wrapper over Tensorflow, which serves a
back-end for Keras, to extract features from ImageNet training images and from query
images uploaded.

3.1.5. Numpy and Scikit-learn. We use them as our tools for vector and matrix computa-
tions, and perform dimension reductions.

3.2. Architecture

3.2.1. Front-end server

(1) It receives a query image (base64 encoding) from users, extracts features from it,
sends the features to feature servers
(2) It receives results (image names and similarity scores) obtained from feature
servers, sorts the results based on scores, and sends top K image names (batch
requests) to the corresponding image servers (image names)
(3) It receives results (images) from image servers, sends them as response to the page
and displays them to users

3.2.2. Feature server

(1) Each feature server stores features of a portion of images
(2) Each feature server receives query image features from the front-end server, com-
pares it to the features of the images it stores, and returns the top K image names
and similarity scores to front-end server

3.2.3. Image server

(1) Each image server stores a portion of images
(2) Each image server receives image names from the front-end server, and gives back
the corresponding images encoded as base64 format

3.2.4. Indexer

(1) Indexer is an off-line component that indexes images
(2) Indexing is done by extracting features from images via Keras and Tensorflow, and
then storing them in a dictionary with image names as keys and features as values

3.2.5. Dimension reducer

(1) It's also an off-line component that embeds original feature space into a much lower
feature space
(2) For our project, we use PCA as our dimension reducer
The interactions between different components can be seen in Figure 2.

4. EVALUATION

Evaluation of our image retrieval system is somewhat subjective, since whether the images retrieved are relevant to the query image is totally dependent on the user's intention and objective. In our evaluation, we define "relevant" as "Either the main objects in the image are the same or the image color and styles are both highly relevant". We use Precision score as our evaluation. We don't use Recall because it's hard to count how many images in our database with 1 million images are relevant to the query image. Thus we cannot use F-score or others dependent on Recall.

We divide the evaluation into two types: In-database Evaluation, which means that the query image is in the database, and Out-database Evaluation, meaning that the query image is not in our database. There 10 students helping us evaluate our system and each of them is required to use 25 images for In-database Evaluation and 25 images for out-database Evaluation. Then we calculated the Precision score for each of them and averaged the scores. Here is our results:

(1) In-database Evaluation with raw features of 2048 dimensions: 0.98
(2) Out-database Evaluation Precision with raw features of 2048 dimensions: 0.85

(1) In-database Evaluation with reduced features of 200 dimensions: 0.93
(2) Out-database Evaluation Precision with reduced features of 200 dimensions: 0.75

From the result, we see that for the query image from the database, our system can accurately retrieve relevant images. We believe that’s partly because the model we used is directly trained for classification on the images in the database. For query images out of the image database, the Precision score is relatively lower, but it's still high enough. It’s lower than in-database score because our database doesn’t include all kinds of objects in the world. We will discuss an improvement on this in the section 5.3. On the other hand, as we reduce the dimensions of features using PCA, the Precision score both decrease in both cases.
Your uploaded image:

Search Result:

Fig. 3. A sample dog image search result

5. DISCUSSION
5.1. Issues encountered, image representation and transferring

Since our project is all about images, the form of image that can be transmitted between web page and front-end server, or between image server to front-end server needs to be seriously considered and carefully handled.

The image uploaded from user needs to be saved in the disk for further operations. From front page to front server, we use JavaScript POST method. The image cannot be simply passed as an argument, and JSON does not support including binary type. Based on those facts, We tried several methods including converting the image to a Format Data type, and played with image types (e.g. PNG, JPEG) for encoding and decoding. Finally, we overcame this issue by extracting the base64 format of images and posting the base64 string representation of images directly to the front-end server. For the front-end server, those representations of images have to be converted as PNG format first (otherwise failed for other formats), and then save to whatever format we want.

Once we got the raw image, we can extract representation codes and parse as an URL argument to other servers. Displaying image then becomes simple since we have overcome the saving image part.
5.2. Success, failures, and feature dimensions

As shown in Evaluation Section, feature dimensions have a big influence on the result of our image search.

Our image feature is a vector with float values. Raw features with 2028 dimensions extracted from the last pooling layer of ResNet50 model would concentrate more on the objects in the image because ResNet50 is trained for object classification and the semantics of features given by the last layer will bias towards objects in the image. In other words, the search results would be more likely to include images with those objects in the original image. For example, searching a complex guitar store in Figure 4 would return the images that contain more guitars than in Figure 5 of vector size 200.

In contrary, when we use PCA to reduce the feature dimensions and if the vector size is not that large but still considerable (like 200), the PCA effect would give a more "general feeling and style" result. Instead of focusing the objects in the images, it will give out the images that have similar environment, background, or colors.

5.3. Future Improvements

(1) In this project we only used the representation codes from the last pooling layer of ResNet50 model. In the future, we would love to try how representation codes from middle layers will affect the performance of our search engine;
(2) As different features offer different effects on retrieval results, we can give multiple options for users to choose from at the same time, such as object-based retrieval, style-based retrieval, color-based retrieval. In that case, we would add more types of features, not only just deep learning features from classification models;

(3) In our project, we simply use Euclidean distance to measure the similarity between the features of two images. But there is no 100 percent guarantee that smaller Euclidean distance means greater similarity. In the future, we may consider adding a similarity module, where the similarity between images are learned during the model training;

(4) This time we only use around 1 million images as our image database. We may consider scaling our architecture to even more images in the future to provide more robust and diversified retrieval;

(5) Our project is motivated by text retrieval. But we can go one step further to extend our framework to video retrieval, not just limited to images. In that case, there will be many more challenges in terms of data storage, video-representation models and search engine architecture.

6. CONCLUSIONS
In this project, we developed a image retrieval system based on image content using deep learning models. Results show that the images retrieved are highly relevant to
the query image. In the future, we intend to do more improvements to make it scale to more images and may extend it to a multimedia retrieval system supporting videos, music audios, etc.

7. REFERENCES IN NEW ACM REFERENCE FORMAT

an article [B. Thomee and Li 2016], an enumerated journal article [J Wan 2014], an article in ECCV [UAI 2011], an article [Lempitsky 2014], an article in ECCV [A Gordo 2016], an article [Vijay Chandrasekhar 2017] an article [Albert Gordo 2016]

ACKNOWLEDGMENTS

The authors would like to thank our course instructor Matt Doherty for giving instructions on our project proposal.

REFERENCES


A. SOME EXAMPLES OF "FAILURES"
Welcome to Image Search

@Author: Haoran Wang, Yuanbin Wang

Please upload your Image (.gif,.jpg,.png):

Your uploaded image:

Search Result:

Fig. 6. Example 1.
Welcome to Image Search

Please upload your image (.gif, .jpg, .png).

Your uploaded image:

Search Result:

Fig. 7. Example 2.

Fig. 8. Example 3.