PyRetri: A PyTorch-based Library for Unsupervised Image Retrieval by Deep Convolutional Neural Networks

Benyi Hu[*]
Xi'an Jiaotong University
hby0906@stu.xjtu.edu.cn

Ren-Jie Song
Megvii Technology
songrenjie@megvii.com

Xiu-Shen Wei[†]
Nanjing University of Science and Technology
weixs.gm@gmail.com

Yazhou Yao
Nanjing University of Science and Technology
yazhou.yao@njust.edu.cn

Xian-Sheng Hua
Alibaba Group
huaxiansheng@gmail.com

Yuehu Liu
Xi'an Jiaotong University
liuyh@mail.xjtu.edu.cn

ABSTRACT

Despite significant progress of applying deep learning methods to the field of content-based image retrieval, there has not been a software library that covers these methods in a unified manner. In order to fill this gap, we introduce PyRetri, an open source library for deep learning based unsupervised image retrieval. The library encapsulates the retrieval process in several stages and provides functionality that covers various prominent methods for each stage. The idea underlying its design is to provide a unified platform for deep learning based image retrieval research, with high usability and extensibility. The project source code, with usage examples, sample data and pre-trained models are available at https://github.com/PyRetri/.

CCS CONCEPTS

• **Software and its engineering** → **Software libraries and repositories**; • **Information systems** → *Top-k retrieval in databases*.

KEYWORDS

Open Source, Image Retrieval, Convolutional Neural Networks

1 INTRODUCTION

Content-based image retrieval (CBIR), which makes use of the representation of visual content to identify relevant images, is one of the fundamental research challenges extensively studied in the multimedia community for decades [16]. Recently, with the prosperity of applying deep learning methods, CBIR has also witnessed the prominence of powerful features of convolutional neural networks. However, the pipeline of deep learning based unsupervised image retrieval is complicated, and empirical configurations used in each stage can have a significant impact on retrieval accuracy. Although there are a wide range of open source implementations released by researchers, codes are not organized in some standardized manners, which may be tedious or confusing for users. Therefore, a high quality and unified framework is essential to keep up the rapid pace of innovation for deep learning based image retrieval researches.

In order to fill this gap, we propose the PyRetri library, an open-source framework that divides deep learning based unsupervised CBIR into several main stages with clear application programming interfaces (APIs). It is a modular Python library based on PyTorch [8] that provides easy-to-use modules to facilitate researchers and engineers in developing unsupervised image retrieval approaches. To the best of our knowledge, this is the first open-source library for unsupervised image retrieval by deep learning.

Towards the goal of providing a high-quality, easy-to-use and easy-to-extend framework, we obey the following principles: (1) **High code quality**. We assure the quality by developing through the peer review process and good software engineering practices. For the reason that generality should not come at the cost of usability, we tackle project maintenance difficulties by providing type hints. Furthermore, the library code is maintained in a clean, consistent style, with class and function names that are descriptive of the underlying functionality. (2) **Human readable configurations**. We utilize YACS [5], a highly human readable configurations management system, to define all the hyper-parameters of methods used in the CBIR pipeline in only one `config` file. Through this serialization format, users can easily manage their own experiments with brief and clear configurations. (3) **Modular design**. Approaches contained in PyRetri do not intend to create module lock-in. Instead, modules are modeled minimal single-function blocks that share an interaction interface, which allows easy plug-ins of user-defined
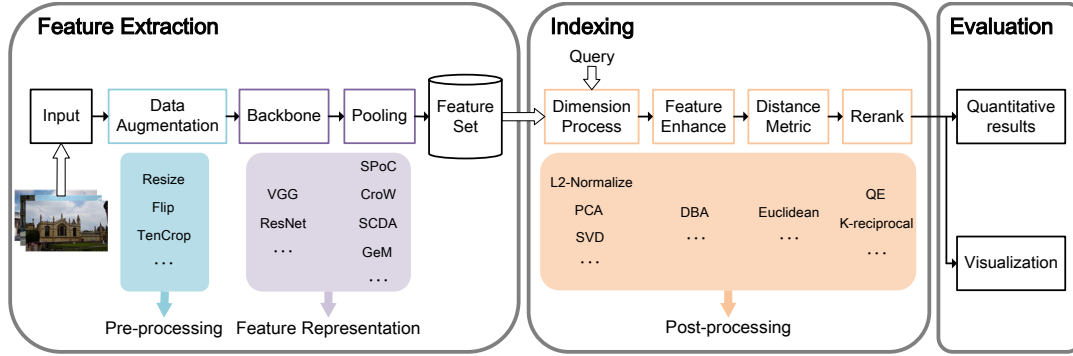
Figure 1: Framework of deep learning based unsupervised image retrieval, illustrated with abstractions in PyRetri.

modules. This spawns a unified environment where developers are able to efficiently explore ideas through high-level module operations, and apply customizations to modules only if necessary.

To summarize, the main contributions of PyRetri are:

- We propose the first open source framework to unify the pipeline of deep learning based unsupervised image retrieval, which is readable and extendable.
- We provide high quality implementations of CBIR algorithms to solve retrieval tasks with emphasis on usability.
- We release reference codes, model zoos and tools of instance-level image retrieval, which can benefit researchers to implement and design their own methods.

2 DESIGN OVERVIEW

The overall architecture of PyRetri is illustrated in Figure 1. In PyRetri, the pipeline of deep learning based unsupervised CBIR is grouped into three crucial modules: feature extraction, indexing and evaluation. In the following, we elaborate these modules.

2.1 Feature Extraction Module

The feature extraction module is utilized to compute the global-level image representation for retrieval with one single network pass. For practical convenience, we first generate a json file to describe the query or gallery datasets, by saving the information of each image such as its path and labels in a list of dictionaries. Given the data augmentation operations and pre-trained models, PyTorch [8] is adopted as the backend and the inference engine to construct a feature extraction pipeline. The output feature is flexibly assigned through a hooking mechanism, whether it is generated by the fully-connected layers or the convolutional layers. Particularly for the convolutional layers, deep descriptors are firstly collected and then aggregated into the global-level representation, which is necessary and discriminative for CBIR [17].

2.2 Indexing Module

The indexing module is the core brick of CBIR, which returns images containing the same content as the query based on the similarity between their image representations. The indexing stages of many retrieval tasks share a similar workflow, where query features and gallery features are projected into a new manifold space and distances are calculated between them. As shown in Figure 1, we

constructed a complete and meticulous indexing pipeline where interfaces are reserved for all indexing stages. Thanks to the modular design, users can plug in their methods easily through these interfaces without changing the core code of the retrieval pipeline.

Similar to the feature extraction module, the retrieval results of each query, a.k.a. neighboring indexes, are add to the json file, which is convenient for the following evaluation process.

2.3 Evaluation Module

The evaluation module is utilized to evaluate the retrieval accuracy and further analyze the retrieval results. We adopt the recall and mean average precision (mAP) as the evaluation metrics, which are widely used in CBIR related tasks. In general, "recall" denotes the ratio of returned true matches to the total number or true matches in the database. "mAP" denotes the average of AP on all queries, which amounts area under the precision-recall curve. Typically, higher recalls and mAPs mean better retrieval accuracy. In our implementations, we provide interfaces for both content-based image retrieval and person re-identification tasks.

In addition, with clear interfaces, we support visualizing retrieval results of a single query image by showing or saving its top-k returned images, which is convenient for failure case analyses.

3 SUPPORTED METHODS

PyRetri contains high-quality implementations of prominent unsupervised CBIR algorithms. As for the supported functionality, we have adopted an object-oriented approach, implementing each algorithm as a class template, while also providing free functions for simpler operations. A list of supported methods is given as follows.

3.1 Pre-processing Methods

- DirectResize (DR): Scaling the height and width of the image to the target size directly.
- PadResize (PR): Scaling the longer side of the image to the target size and filling the remaining pixels with the mean values of ImageNet.
- ShorterResize (SR): Scaling the shorter side of the image to the target size.
- TwoFlip (TF): Returning the original image and the corresponding horizontally flipped image.

- CenterCrop (CC): Cropping the image from its center region according to the given size.
- TenCrop (TC): Cropping the original image and the flipping image from up down left right and center, respectively.

3.2 Feature Representation Methods
- GAP: Global average pooling.
- GMP: Global max pooling.
- R-MAC [14]: Calculating feature vectors based on the regional maximum activation of convolutions.
- SPoC [2]: Assigning larger weights to the central descriptors during aggregation.
- CroW [7]: A weighted pooling method for both spatial- and channel-wise.
- SCDA [17]: Keeping useful deep descriptors based on the summation of feature map activations.
- GeM [11]: Exploiting the generalized mean to reserve the information of each channel.
- PWA [19]: Aggregating the regional representations weighted by the selected part detectors' output.
- PCB [13]: Outputting a convolutional descriptor consisting of several part-level features.

3.3 Post-precessing Methods
- SVD [6]: Reducing feature dimension through singular value decomposition of matrix.
- PCA [18]: Projecting high-dimensional features into fewer informative dimensions.
- DBA [1]: Every feature in the database is replaced with a weighted sum of the point's own value and those of its top k nearest neighbors (k-NN).
- QE [3]: Combining the retrieved top-k nearest neighbors with the original query and doing another retrieval.
- k-reciprocal [22]: Encoding k-reciprocal nearest neighbors to enhance the accuracy of retrieval.

4 CONFIGURATION SEARCH TOOL
Since different algorithms used in each stage might have a significant impact on retrieval accuracy, we present the configuration search tool to help users to find the optimal retrieval configuration with various hyper-parameters.

As the same coding style as PyRetri, our configuration search tool is easy to read and deploy. The tool consists two components: the search space and search script. The search space is defined by the users through adding methods with hyper-parameters to a specified dict. Then, the search script completes the retrieval process based on all the configurations within the search space in an exhaustive way, saving the results automatically.

Moreover, in order to help users analyze retrieval results easily, we also provide scripts to convert the results file into the csv format or filter the retrieval results according to the given key words.

5 APPLICATIONS AND EVALUATIONS
Recently, deep learning based approaches are widely explored for unsupervised image retrieval, which have achieved satisfactory results and have been successfully applied to diverse multimedia

and computer vision tasks like content-based image retrieval and person re-identification, etc. We validate the effectiveness of PyRetri on the two tasks respectively through experiments on several benchmark datasets.

5.1 Benchmark Datasets
- Oxford5k [9] collects crawling images from Flickr using the names of 11 different landmarks in Oxford, which is a representative landmark retrieval task.
- CUB-200-2011 [15] contains photos of 200 bird species, which represents fine-grained image retrieval.
- Indoor [10] contains indoor scene images with 67 categories, representing for the scene retrieval/recognition task.
- Caltech101 [4] consists pictures of objects belonging to 101 categories, standing for the generic image retrieval task.
- Market-1501 [20] contains images taken on the Tsinghua campus under six camera viewpoints, which is the benchmark dataset for person re-identification.
- DukeMTMC-reID [12] contains images captured by eight cameras, which is a more challenging person Re-ID dataset.

5.2 Content-Based Image Retrieval
For the configurations of content-based image retrieval, we pick up the following search factors for evaluation: data augmentation, backbone, aggregation methods and dimension process/reduction. The data augmentation operations include SR+CC, PR and DR (cf. Sec. 3.1), aiming at finding the relationship between image integrity and the retrieval accuracy. All the aggregation methods are added to the search space in order to get comprehensive analyses. In addition, popularly used dimension process/reduction approaches are adopted for searching, such as ℓ_2-normalization (ℓ_2), PCA with whitening (PCA_w) or PCA without whitening (PCA) and SVD with whitening (SVD_w) or without whitening (SVD). After getting the top-3 best retrieval accuracy w.r.t. the searched configurations on each dataset, we further search for the optimal feature enhance and rerank operations.

As reported in Table 1, proper retrieval configurations bring a significant improvement on the retrieval accuracy on these CBIR benchmark datasets. Also, the inference speed of each retrieval stage is shown in Table 2. The averaged inference time per image by our PyRetri is less than 8ms.

5.3 Person Re-Identification
In addition to the general image retrieval task, we also apply PyRetri to person re-identification, which owns the retrieval workflow.

Since the model for Re-ID tasks has been trained on the target dataset, we do not search for feature extraction operations and just evaluate the accuracy of PyRetri based on the open source pre-trained models. Since the developer does not give the model trained on DukeMTMC-reID, we train a model on it by ourselves by employing the widely used code [21], which is utilized as the baseline of the Re-ID experiments.

As shown in Table 3, our re-ID results are able to be the same as the results reported in the original implementation, which proves that our PyRetri is reliable. More importantly, by applying the retrieval configurations provided by PyRetri, our results outperform

Table 1: Top-3 retrieval accuracy w.r.t. the corresponding searched configurations and the baseline of each dataset. Compared with the baseline consisting of default configurations, proper retrieval configurations can outperform it by a large margin.

Dataset	Config.	Pre-trained data	Data aug	Backbone	Output layer	Aggregation	Dimension process	Feature enhance	Re-rank	mAP
Oxford5k	Search Configs.	ImageNet+Places	SR+CC	VGG16	pool5	GAP	$\ell_2+SVD_w+\ell_2$	–	k-reciprocal	**72.9 (+26.6)**
		ImageNet+Places	DR	Res50	pool5	SPoC	$\ell_2+PCA_w+\ell_2$	–	k-reciprocal	72.4
		ImageNet+Places	SR+CC	VGG16	pool5	GAP	$\ell_2+PCA_w+\ell_2$	–	k-reciprocal	72.1
	Baseline	ImageNet	SR+CC	Res50	pool5	GAP	–	–	–	46.3
CUB-200	Search Configs.	ImageNet	SR+CC	Res50	pool5	SCDA	$\ell_2+PCA+\ell_2$	-	k-reciprocal	**38.9 (+21.0)**
		ImageNet	SR+CC	Res50	pool5	GeM	$\ell_2+PCA+\ell_2$	–	k-reciprocal	37.3
		ImageNet	SR+CC	Res50	pool5	GMP	$\ell_2+PCA+\ell_2$	–	k-reciprocal	37.2
	Baseline	ImageNet	SR+CC	Res50	pool5	GAP	–	–	–	17.9
Indoor	Search Configs.	Places	DR	Res50	pool5	CroW	$\ell_2+PCA+\ell_2$	DBA	QE	**63.7 (+39.8)**
		Places	DR	Res50	pool5	GAP	$\ell_2+PCA+\ell_2$	DBA	QE	63.5
		Places	DR	Res50	pool5	GeM	$\ell_2+PCA+\ell_2$	DBA	QE	63.2
	Baseline	ImageNet	SR+CC	Res50	pool5	GAP	–	–	–	23.9
Caltech101	Search Configs.	ImageNet	PR	Res50	pool5	GMP	$\ell_2+PCA+\ell_2$	DBA	QE+k-reciprocal	**86.4 (+19.3)**
		ImageNet	PR	Res50	pool5	GeM	$\ell_2+PCA+\ell_2$	DBA	QE+k-reciprocal	86.1
		ImageNet	PR	Res50	pool5	SCDA	$\ell_2+PCA+\ell_2$	DBA	QE+k-reciprocal	86.1
	Baseline	ImageNet	SR+CC	Res50	pool5	GAP	–	–	–	67.1

Table 2: Inference speed comparisons of each retrieval stage.

Stage	♯ images	Resolution	Backbone	Avg. time (ms/img)
Extraction	100	224×224	VGG16	7.16
			Res50	7.72
Indexing	Query set: 100 Gallery set: 100	224×224	VGG16	0.02
			Res50	0.03

Table 3: Re-ID accuracy of benchmark datasets. Our implementation achieves accuracy on par with the reported results. With the optimal configuration provided by PyRetri, the results significantly boost.

Dataset	Implementations	mAP	Recall@1
Market-1501	Referenced impl. [21]	71.6	88.8
	Ours	71.6	88.8
	Ours w. optimal config.	**84.8**	**90.4**
DukeMTMC-reID	Referenced impl. [21]	–	–
	Ours	62.5	80.4
	Ours w. optimal config.	**78.3**	**84.2**

the baseline by a large margin, justifying the practical effectiveness of the library.

6 AVALIABILITY

PyRetri is released under the license of Apache 2.0 and its source code is openly available at: **https://github.com/PyRetri/PyRetri**. We also provide extensive documentations and sampled projects for it. Contributions from the open-source community are welcome, via the GitHub issues/pull request mechanisms.

7 CONCLUSIONS

In this project, we have developed PyRetri, a software library that focuses on deep learning based unsupervised image retrieval. Our library unifies the pipeline of CBIR and provides convenient interfaces for each retrieval stage, which can be easily adopted for various multimedia application scenarios. With modular designs and object-oriented implementations, PyRetri is easy-to-use and easy-to-extend, which is suitable to be a codebase for other researchers. We hope that PyRetri can provide unique convenience to the deep learning and CBIR research community.

REFERENCES

[1] R. Arandjelović and A. Zisserman. 2012. Three things everyone should know to improve object retrieval. In *CVPR*. 2911–2918.
[2] A. Babenko and V. Lempitsky. 2015. Aggregating deep convolutional features for image retrieval. In *ICCV*. 1269–1277.
[3] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. 2007. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*. 1–8.
[4] L. Fei-Fei, R. Fergus, and P. Perona. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR workshop*. 178–186.
[5] R. Girshick. 2018. YACS. Website. https://github.com/rbgirshick/yacs.
[6] G.H. Golub and C. Reinsch. 1970. Singular value decomposition and least squares solutions. *Numer. Math.* 14, 5 (1970), 403–420.
[7] Y. Kalantidis, C. Mellina, and S. Osindero. 2016. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV workshop*. 685–701.
[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kääpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 8024–8035.
[9] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*. 1–8.
[10] A. Quattoni and A. Torralba. 2009. Recognizing indoor scenes. In *CVPR*. 413–420.
[11] F. Radenović, G. Tolias, and O. Chum. 2019. Fine-tuning CNN image retrieval with no human annotation. *IEEE TPAMI* 41, 7 (2019), 1655–1668.
[12] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV workshop*. 17–35.
[13] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang. 2018. Beyond part models: Person retrieval with refined part pooling. In *ECCV*. 501–518.
[14] G. Tolias, R. Sicre, and H. Jégou. 2016. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*. 1–12.
[15] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. *The Caltech-UCSD Birds-200-2011 Dataset*. Technical Report CNS-TR-2011-001. California Institute of Technology.
[16] J. Wan, D. Wang, S. C.-H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. 2014. Deep Learning for Content-Based Image Retrieval: A Comprehensive Study. In *ACM, MM*. 157–166.
[17] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou. 2017. Selective convolutional descriptor aggregation for fine-grained image retrieval. *IEEE TIP* 26, 6 (2017), 2868–2881.
[18] S. Wold, K. Esbensen, and P. Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.
[19] J. Xu, C. Shi, C. Qi, C. Wang, and B. Xiao. 2018. Unsupervised part-based weighting aggregation of deep convolutional features for image retrieval. In *AAAI*. 7436–7443.
[20] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. 2015. Scalable person re-identification: A benchmark. In *ICCV*. 1116–1124.
[21] Z. Zheng. 2018. https://github.com/layumi/Person_reID_baseline_pytorch.
[22] Z. Zhong, L. Zheng, D. Cao, and S. Li. 2017. Re-ranking person re-identification with k-reciprocal encoding. In *CVPR*. 1318–1327.