Learning Local Similarity with Spatial Relations for Object Retrieval

Zhenfang Chen
The University of Hong Kong, Hong Kong
zfchen@cs.hku.hk

Wayne Zhang
SenseTime Research, Hong Kong
wayne.zhang@sensetime.com

Zhanghui Kuang
SenseTime Research, Hong Kong
kuangzhanghui@sensetime.com

Kwan-Yee K. Wong
The University of Hong Kong, Hong Kong
kykwong@cs.hku.hk

ABSTRACT

Many state-of-the-art object retrieval algorithms aggregate activations of convolutional neural networks into a holistic compact feature, and utilize global similarity for an efficient nearest neighbor search. However, holistic features are often insufficient for representing small objects of interest in gallery images, and global similarity drops most of the spatial relations in the images. In this paper, we propose an end-to-end local similarity learning framework to tackle these problems. By applying a correlation layer to the locally aggregated features, we compute a local similarity that can not only handle small objects, but also capture spatial relations between the query and gallery images. We further reduce the memory and storage footprints of our framework by quantizing local features. Our model can be trained using only synthetic data, and achieve competitive performance. Extensive experiments on challenging benchmarks demonstrate that our local similarity learning framework outperforms previous global similarity based methods.

CCS CONCEPTS

• Information systems → Image search.

KEYWORDS

Object Retrieval, Local Spatial Relations, Computer Vision

ACM Reference Format:

1 INTRODUCTION

Object retrieval has been a fundamental and popular research topic in computer vision. It aims at retrieving images containing objects same as the query image from a large database. Classical methods often represent each image by one or more descriptors, and formulate object retrieval as a nearest neighbor search in the descriptor space [2, 6, 7, 16, 22, 25–28, 41]. Recently, deep convolutional features have been extensively explored for image representation, and they have shown excellent performance over conventional features in many vision tasks. Many efforts have therefore been devoted to designing and learning good holistic image representations based on deep convolutional features, and object retrieval

Figure 1: Illustration of spatial relations encoded in the correlation volume. In (a), we show how the spatial relations are encoded in the correlation volume using a toy example. The query contains 9 local regions (i.e., A, B,...,I). For each local region, it has a best-matching region in the gallery (indicated by the same character) which gives the maximum activation along the depth channel. In the green rectangle, we show a positive gallery which is an affine transformation of the query. Since affine transformation is linear, we can see the coordinates of all the maximum activations fall on a plane. In the red rectangle, we show a gallery image which contains the same local regions as the query but with different spatial relations, and we see the coordinates of the maximum activations distribute randomly. In (b), we show a real example with feature maps of the query and the gallery pooled to 7 × 7 and 15 × 15, respectively. ResNet101 is used as the feature extractor and only the strongest 20% activations are shown. We observe that the coordinates of most activations fall on a plane in the positive correlation volume while activations randomly distribute in the negative case. (best viewed on screen). See section 3.2 for further explanation.
2 RELATED WORKS

Object retrieval has been studied for more than a decade. A comprehensive survey, which categorizes existing methods into local feature based methods and global feature based methods, can be found in [48]. In [41], Sivic and Zisserman proposed the Bag-of-the-Word (BoW) model which encodes a set of local invariant features (e.g., SIFT [21]) in a sparse feature vector for image retrieval. Following this seminal work, researchers introduced various components, such as large visual codebooks [3, 22], spatial verification [25, 28] and query expansion [6, 7], into the pipeline to improve the retrieval performance. Other classical research works, such as Fisher Vector [26, 27] and VLAD [2, 16], focused on designing schemes for aggregating local features into compact global features.

In the past few years, convolutional neural networks (CNNs) were widely adopted for the object retrieval task, and they demonstrated extraordinary performance. Many of such works [4, 5, 10, 11, 18, 33–35, 42, 45] focused on designing and learning holistic compact features. Early works [4, 33] extracted features directly from the fully connected layers of a pre-trained CNN, while recent works commonly aggregated regional features from the convolutional layers [1, 42, 45]. Researchers also found that fine-tuning CNNs on data similar to the target task can significantly boost the retrieval performance [4, 10, 11, 32]. These methods typically adopted a nearest neighbor search based on global similarities of the holistic compact features. Although searching based on global similarity is efficient, it does not perform well under more challenging conditions, such as complicated clutter, large occlusion and variations in viewpoints. This is because global similarity is ineffective in representing similarities between local regions as it drops the spatial relations of local regions in the images.

In [23], Noh et al. proposed a deep local feature image retrieval method which utilizes an attention mechanism to extract local features from a query image and performs nearest neighbor search for each local feature. Their method then aggregates all the matches per gallery image. Iscen et al. [15] proposed a diffusion mechanism to capture the manifold in the local feature space. The diffusion
is carried out on descriptors of overlapping image regions rather than on a holistic image descriptor. The use of local features has boosted the retrieval performance significantly in these approaches. However, their local feature based similarities are not defined explicitly, and their pipelines are rather complicated. For instance, the regional diffusion method [15] requires storing and computing on a huge graph. In contrast, we propose a simple and effective end-to-end local similarity learning network for object retrieval.

CNNs have also demonstrated great success in image matching. Many research works [12, 43, 46, 47] focused on part of or the whole pipeline for detecting local feature keypoints and comparing local features. Recently, Rocco et al. [37] proposed a CNN architecture trained on synthetically generated images for predicting an affine or thin-plate-spline transformation for image matching. However, all these methods were not originally designed for the object retrieval task. In this paper, we cast the problem of object retrieval as object localization (i.e., checking whether the query object exists in the gallery image and finding its position), and supervise our proposed end-to-end trainable network with an object localization loss. We also demonstrate how our object retrieval algorithm can be scaled up to handle large-scale datasets.

3 LOCAL SIMILARITY WITH SPATIAL RELATIONS

In this section, we first briefly describe the working principle of existing methods that are based on global similarity, and the problems associated with them. This leads us to explore taking the spatial relations of local regions into account in computing local similarity for object retrieval. Specifically, we propose to compute similarities between local regions of the query and gallery images using correlation, and rearrange the results into a correlation volume that allows easy indexing the similarities by spatial indices of the local regions. We prove that by preserving the spatial relations of local regions in both the query and gallery images, maximum responses in this correlation volume should lie in a specific subspace. Based on this proof, we propose using a small CNN to learn the existence and location of the object of interest from the correlation volume.

3.1 Deficiency of Global Similarity

Many recent CNN-based object retrieval algorithms follow the global R-MAC pipeline [42] which subdivides an image into a grid of rectangular regions, extracts local features from the regions, and aggregates these regional features to a holistic image representation [5, 10, 11, 14, 18, 32]. The R-MAC extraction process can be summarized as follows. First, activation features are extracted from the convolutional layers of a pre-trained network. These activation features are then max-pooled in each region. The pooled regional features are independently l2-normalized, whitened with PCA and l2-normalized again. Finally, these normalized regional features are sum-aggregated and l2-normalized to produce a holistic descriptor. A global similarity between two images can then be computed as the dot-product of their holistic descriptors.

Let \( f_q(x_q) \) be the normalized D-dimensional regional feature with spatial index \( x_q \in \mathbb{R}^2 \) for the query image. Similarly, let \( f_p(x_p) \) be the normalized regional feature with spatial index \( x_p \in \mathbb{R}^2 \) for the gallery image. The global R-MAC descriptors for the query and gallery images can be written as \( g_q = \sum_{x_q} f_q(x_q) \) and \( g_p = \sum_{x_p} f_p(x_p) \). The global similarity between the query and gallery images is then given by

\[
S_{R-MAC} = g_q^T g_p = \sum_{x_q, x_p} f_q(x_q)^T f_p(x_p). \tag{1}
\]

It follows from (1) that the global R-MAC similarity can be interpreted as a cross matching between all regions of the query and gallery images. This formulation has two potential problems. First, each pair of regions contributes equally to the final similarity, and this makes the contributions of the true corresponding regions less significant (i.e., a low signal-to-noise ratio). Second, the sum-aggregation is an orderless operation over the set of regional features, and is incapable of preserving the spatial relations of local regions in the image. The seminal work R-Match [34] solves the first problem by adopting a winner-take-all approach. It greedily finds the best matched local region in the gallery image for each local region in the query image, and then sum-aggregates the similarities of the best matches into the final similarity. Formally, the R-Match similarity is given by

\[
S_{R-Match} = \max_{x_q} \left\{ f_p(x_q)^T f_q(x_q) \right\}. \tag{2}
\]

Note that R-Match does not take the spatial relations of local regions into account in computing the local similarity. It would therefore output an incorrect high similarity when the gallery image contains local regions which are similar to those of the query image but have a different spatial composition (see the shuffle case in Fig. 1a).

3.2 Preserving Spatial Relations of Local Regions

In order to allow us to take the spatial relations of local regions into account in computing local similarity, we propose to compute similarities between local regions of the query and gallery images using correlation, and rearrange the results into a correlation volume that indices the similarities by spatial locations. Let \( f_q \in \mathbb{R}^{W_q \times H_q \times D} \) denote a 2D map of D-dimensional regional features for the query image, where \( W_q \times H_q \) is the spatial resolution of the subdivision grid for the query image. Similarly, let \( f_p \in \mathbb{R}^{W_p \times H_p \times D} \) denote a 2D map of D-dimensional regional features for the gallery image, where \( W_p \times H_p \) is the spatial resolution of the subdivision grid for the gallery image. We define the similarity between the local region of the query image with spatial index \((w_q, h_q)\) and the local region of the gallery image with spatial index \((w_g, h_g)\) as

\[
s(w_q, h_q, w_g, h_g) = f_p(w_g, h_g)^T f_q(w_q, h_q). \tag{3}
\]

By considering the similarities between all regions of the query and gallery images, we produce a tensor of similarities with a dimension of \( W_q \times H_q \times W_g \times H_g \). We rearrange the last two dimensions related to the query image into a single dimension referred to as the depth channel, and obtain a volume of similarities with a dimension of \( W_g \times H_q \times D' \) where \( D' = W_q \times H_q \times D \). We can interpret this volume as a 2D map (i.e., \( W_g \times H_q \)) of \( D' \)-dimensional correlation vectors \( f_p(w_g, h_g) \), where each correlation vector \( f_p(w_g, h_g) \) encodes how...
well the local region of the gallery image with spatial index \((w_q, h_q)\) matches with each local region of the query image.

The above computation can be conveniently implemented with a correlation layer which is first used in CNN-based stereo matching and optical flow estimation [8, 20]. Note that the correlation volume generated by the correlation layer stores the similarities in such a way that allows easy indexing the similarities by spatial indices of the local regions. Next, we are going to show that by preserving the spatial relations of local regions in both the query and gallery images, maximum responses in this correlation volume should lie in a specific subspace. Consider the simple case where the query and the gallery images are identical. Let \(d' = [1, W_q \times H_q]\) denote the index of the maximum response for the correlation vector \(f_c(w_q, h_q)\).

Since the query and gallery images are identical, maximum response should happen when the local regions of the query and gallery images have the same spatial index (i.e., \((w_q, h_q) = (w_y, h_y)\)). Hence, we have

\[
d' = K[w_y \ h_y]^{T} - W_y, \tag{4}
\]

where \(K = [1 \ W_y]\). Now consider the case where there exists a transformation between the query and gallery images. Let \(\Phi\) be a function that maps each local region of the gallery image to a corresponding local region of the query image, i.e.,

\[
[w_y \ h_y]^{T} = \Phi[w_y \ h_y]^{T}. \tag{5}
\]

Substituting (5) into (4) gives

\[
d' = K\Phi([w_y \ h_y]^{T}) - W_y, \tag{6}
\]

which defines the subspace in which the maximum responses corresponding to true matching of local regions between the query and gallery images should lie. It is easy to see that if the mapping function \(\Phi\) is linear (e.g., translation, rotation, scaling, affine transformation), this subspace will become a plane. We illustrate the existence of such a subspace for maximum responses in Fig. 1 using both toy and real examples. It can be observed that maximum responses lie on a plane for positive gallery images, whereas there is a more ‘random’ distribution of maximum responses for negative gallery images.

### 3.3 Learning Local Similarity with Object Localization

From the analysis above, we find that the correlation volume \(f_c\) not only encodes the pairwise similarity between all the local region pairs, but also captures the spatial relations. Since our goal is to estimate the similarity between the query and the gallery image, we design a lightweight CNN \(\mathcal{F}\) to classify the pattern encoded in the correlation volume. The predicted confidence for a positive pair is used as the estimated similarity between the query and the gallery. Formally,

\[
S_{local} = \mathcal{F}(f_c). \tag{7}
\]

As the correlation volume encodes information of whether the query exists in the gallery as well as the location(s) of the query, we model the retrieval problem as object localization on the correlation volume. Following the spirit of Faster R-CNN [36], we apply classification and localization in parallel. Formally, during training, we define a multi-task loss as

\[
\mathcal{L}(y, \tilde{y}, l, \tilde{l}) = \mathcal{L}_{cls}(y, \tilde{y}) + \lambda \mathcal{L}_{loc}(l, \tilde{l}), \tag{8}
\]

where \(y\) is the probability of object existence predicted by the convolutional subnetwork, \(\tilde{y}\) is the ground truth label of object existence (\(\tilde{y} = 1\) if the query object exists in the gallery image, and \(\tilde{y} = 0\) otherwise), \(l\) and \(\tilde{l}\) denote the predicted and ground truth normalized bounding boxes of the query object in the gallery image, respectively. \(\lambda\) is simply a balancing weight which is set to 0.05 in our implementation.

The classification loss \(\mathcal{L}_{cls}(y, \tilde{y})\) is computed based on cross entropy loss. We use the classification loss rather than the triplet loss that is used in [10, 11], because the objective of the proposed CNN \(\mathcal{F}\) is to classify the correlation volume, not to differentiate between the features from two images.

The localization loss \(\mathcal{L}_{loc}(l, \tilde{l})\) is computed as smooth \(l_1\) loss as defined in [36], with two major differences. First, the localization loss is normalized by the number of positive query gallery pairs in a mini-batch, i.e., \(N_{loc} = \sum_{i=1}^{N} y_i\) where \(N\) is the size of a mini-batch, but not the number of positive region proposals as in [36]. Second, the bounding box is encoded and normalized with the height and width of the input image as the reference rectangle but not region proposals or default boxes as in [36]. We emphasize that our task is object retrieval but not object detection. We assume there is at most one query object in one gallery image during training. This is actually true for our training data. For testing, we have no such limitation.

After training, the probability of object existence predicted by the CNN \(\mathcal{F}\) can be used directly as the similarity between the query and gallery images. Since the correlation layer encodes the matching between local regions, we name our similarity as local similarity, in contrast to global similarity which compares holistic image representations.

To demonstrate the effectiveness of the proposed loss, we compare the performance of models trained with different loss functions in Table 1. We can see that the classification loss outperforms the triplet loss, and the localization loss further improves the performance since it utilizes additional information for supervision.

By making use of the above multi-task loss, we find that the CNN \(\mathcal{F}\) can learn to mine the spatial relation consistency. To apprehend its capability, we apply Grad-CAM [39] to visualize the convolutional feature maps. Fig. 2 illustrates that the CNN \(\mathcal{F}\) gradually focuses on the region with spatial relation consistency and discards those without it.

### 3.4 Local Similarity Learning CNN

Our local similarity learning CNN consists of two major components, namely a feature extraction subnetwork and an object localization subnetwork. Fig. 3 shows the overall architecture of the proposed framework.
Feature extraction subnetwork. Theoretically, any kind of architectures can be used as the backbone for our feature extraction subnetwork. For simplicity, we take ResNet-101 as the backbone for analysis here. To extract better representations for small objects, we do not resize the input image before feeding it into the feature extraction subnetwork. Hence, our feature extraction subnetwork produces feature maps of different sizes depending on the input image sizes. To handle the differences in size of the feature maps, on the last convolution of ResNet-101, we use one ROI-pooling layer (with ROI being the whole image) to pool the feature map of the gallery image to a fixed size of $15 \times 15 \times 2048$, and that of the query image to $7 \times 7 \times 2048$. Note that the feature map of the query image has a smaller size than that of the gallery image. Such a design is based on the fact that the (cropped) query image typically contains only the object of interest, whereas the object of interest often occupies only a small region of the gallery image. A small query feature map results in a smaller number of depth channels in the correlation volume and hence a faster similarity computation. We further $L_2$-normalize the pooled feature maps at each location to avoid magnitude unbalance between the query and gallery feature maps before feeding them into the object localization subnetwork.

Object localization subnetwork. With a correlation layer taking the query and gallery feature maps as input, our object localization subnetwork utilizes a lightweight CNN to predict whether the query exists in the gallery image with a localization auxiliary task. It has only 5 convolutional layers with a small number of channels, and two small fully connected layers. Each convolutional layer is followed by an instance normalization layer and a ReLU layer. The detailed configuration of the object localization subnetwork can be seen in Table 2.

Our object localization subnetwork is lightweight. The total runtime for retrieving a query is the sum of the runtime for extracting query features and the runtime for predicting similarity. The runtime of our feature extraction subnetwork is similar to that of the global feature extractor [11], both taking about 0.09s on average for an image of size 1024 $\times$ 768 (typical size for images in Oxford building and Paris datasets) using GPU (e.g., Titan X). Our object location subnetwork can make a prediction in less than 0.07ms, achieving about 16,000 frames per second, which is less than 0.1% of the runtime for feature extraction.

Table 2: Detailed configuration of the proposed object localization subnetwork.

<table>
<thead>
<tr>
<th>Type</th>
<th>Filter Shape</th>
<th>Input Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv Block</td>
<td>[3 × 3, 128]×3</td>
<td>15×15×49</td>
</tr>
<tr>
<td>Max Pool</td>
<td>Pool 2×2</td>
<td>15×15×128</td>
</tr>
<tr>
<td>Conv Block</td>
<td>[3×3, 64]×2</td>
<td>7×7×128</td>
</tr>
<tr>
<td>Max Pool</td>
<td>Pool 2×2</td>
<td>7×7×128</td>
</tr>
<tr>
<td>FC</td>
<td>576×2</td>
<td>3×3×64</td>
</tr>
</tbody>
</table>

3.5 Scaling Up for Efficient Object Retrieval

Scalable storage. We pre-compute and store the outputs of the feature extraction subnetwork for the gallery images for saving time and computational resource. Given a gallery image, its feature map has a dimension of $W_d \times H_d \times D$ (where $D$ denotes the number of channels). For a large gallery set, it would mean a high memory and storage cost. In order to reduce the memory and storage footprint, each $D$-dimensional feature vector is first subdivided into $M$ sub-vectors and then $k$-means clustering is run on the uncompressed sub-vectors to generate $C$ centers for each subpart. Finally, each subvector is approximated by one of the $C$ centers. In this way, each feature vector can be approximated using only $M \times \log_2(C)$ bits. Typically, $C$ is set to 256, and $M$ is set to 128 in our case. The total storage can therefore be reduced by 64 times. Experimental results show that, after quantization, the retrieval performance decreases only by 1.2% and 0.5%, respectively on Oxf5k [28] and Par6k [30] benchmarks. It suggests that the proposed object localization subnetwork is robust against small perturbation of the input, and has a good generalization ability. Fine-tuning the object localization subnetwork after quantization can further improve its performance consistently. A fine-tuned subnetwork with $M = 128$ (i.e., 64 times compression) has negligible drops (0.9% on Oxf5k and 0.2% on Par6k) on performance as its uncompressed counterpart on both benchmarks. More detailed results can be found in Section 4.4.

Scalable search. For large-scale databases, we extract a global feature on top of the convolutional layer to accelerate the whole retrieval process. Specifically, we first use the global feature to filter out most irrelevant images, and get a short list of $K$ gallery images. The proposed local similarity is then computed only between the query and the shortlisted images. Theoretically, any global feature can be used to accelerate the search process. To harvest the power
we automatically remove those instances whose areas are smaller.

Titan X), the separated 10 convolutional layers add about 0.004 s.

gallery images and the cropped regions of interest as query images.

seen in Fig 4. During training, we use the whole synthetic images as

blending to further increase the diversity of the training dataset.

photo-metric distortion and random cropping, is carried out before

ages. Data augmentation, including rotation, scale transformation,

the query objects into any random position of the background im-

because it provides diverse backgrounds. We paste

in about 4,000 objects.

sample 5,000 images from COCO validation set [19]. For each image,

masks for the object instances. We use the masks of object instances

summarized as follows.

scalable way. The procedure for generating synthetic data can be

for instance detection [9], we generated a large set of training

Previous research [10, 11, 32] proposed to collect a large real dataset

from the web and label training images with sophisticated methods

for image retrieval. Inspired by recent cutting-and-pasting approach

for instance detection [9], we generated a large set of training

images with bounding boxes annotations in a simpler and more

scalable way. The procedure for generating synthetic data can be

summarized as follows.

Cutting. To cut objects from images, we first need the instance

masks for the object instances. We use the masks of object instances

provided by the annotations of COCO dataset [19]. We randomly

sample 5,000 images from COCO validation set [19]. For each image,

we automatically remove those instances whose areas are smaller

than 80 × 80 and keep only one instance mask per image, resulting

in about 4,000 objects.

Pasting. We use images from ImageNet dataset [38] as the back-

ground dataset because it provides diverse backgrounds. We paste

the query objects into any random position of the background im-

ages. Data augmentation, including rotation, scale transformation,

photo-metric distortion and random cropping, is carried out before

blending to further increase the diversity of the training dataset.

Samples of the original images, masks and synthetic data can be

seen in Fig 4. During training, we use the whole synthetic images as
gallery images and the cropped regions of interest as query images.

of deep features and reduce computational cost, we extract the pop-

ular and effective R-MAC [42] features. Directly encoding R-MAC

on activations of the last convolutional layer of ResNet-101 resulted

in poor performance of the local similarity learning task. This is

reasonable since global similarity and local similarity need different

semantic representation. We gradually reduce the shared layers

between R-MAC and the proposed feature extraction subnetwork,

and experimentally find that sharing the first 91 layers can achieve

a good trade-off between effectiveness and efficiency. Therefore, in

our final design, R-MAC and the proposed feature extraction sub-

network each have one separated small branch of 10 convolutional

layers (92 ~ 101 layers in ResNet-101). When testing on a GPU (e.g.,

Titan X), the separated 10 convolutional layers add about 0.004s

for the total feature extraction and take up 5% of the total feature

extraction time. Although the global feature R-MAC is introduced

here for efficient search initially, we find that the performance of

the proposed method can be boosted further by combining local

similarity with the global similarity based on R-MAC.

3.6 Training with Synthetic Data.

Previous research [10, 11, 32] proposed to collect a large real dataset

from the web and label training images with sophisticated methods

for image retrieval. Inspired by recent cutting-and-pasting approach

for instance detection [9], we generated a large set of training

images with bounding boxes annotations in a simpler and more

scalable way. The procedure for generating synthetic data can be

summarized as follows.

Cutting. To cut objects from images, we first need the instance

masks for the object instances. We use the masks of object instances

provided by the annotations of COCO dataset [19]. We randomly

sample 5,000 images from COCO validation set [19]. For each image,

we automatically remove those instances whose areas are smaller

than 80 × 80 and keep only one instance mask per image, resulting

in about 4,000 objects.

Pasting. We use images from ImageNet dataset [38] as the back-

ground dataset because it provides diverse backgrounds. We paste

the query objects into any random position of the background im-

ages. Data augmentation, including rotation, scale transformation,

photo-metric distortion and random cropping, is carried out before

blending to further increase the diversity of the training dataset.

Samples of the original images, masks and synthetic data can be

seen in Fig 4. During training, we use the whole synthetic images as
gallery images and the cropped regions of interest as query images.

4 EXPERIMENTS

In this section, we demonstrate the performance of our proposed

object retrieval algorithm with extensive experiments. First, we

show that our pipeline can be trained effectively with only syn-

thetic data and achieve competitive performance. We then show the

effectiveness of local similarity. We also evaluate the techniques of

scalable storage and scalable search. Finally, we compare our object

retrieval algorithm with state-of-the-art methods using global simi-

larity (e.g., DIR [10, 11]) and regional features (e.g., R-Match [34]

and DELF [23]).

4.1 Experimental Settings

Evaluation datasets and criterion. We evaluate our methods

on popular object retrieval datasets, namely Oxford Building [28],


[30] dataset, composed of 5,063 and 6,412 images, are referred to as

Oxf5k and Par6k, respectively. Besides, 100k Flickr images [28] are

added to these two datasets to form Oxf105k and Par106k datasets

evaluation at a larger scale. To better show the effectiveness

of the proposed method to capture the spatial relations, we also

create new datasets by adding more challenging synthetic negative

distractors into Oxf5k and Par6k. Specifically, every positive gallery

image is first subdivided into a grid of 10 × 10 and then we shuffle

these 100 patches randomly. We finally manually check that the

region of interest is fully destroyed. Fig 5 shows some examples

of synthetic distractors, which contain local patches similar to the

query images but with a total different spatial arrangement. For

each positive gallery image, we generate 3 synthetic distractors

and combine them with the original Oxf5k and Par6k to form two

new testing sets which contain 6,764 and 11,782 images, respecti-

vely. We refer to these two new datasets as OxfShf and ParShf,

respectively. As for INSTRE [44], it contains 250 different objects

and include more variations such as scales, rotations and occlusions

than Oxford and Paris, which make it more challenging. Retrieval

performance is measured in terms of mean average precision (mAP)

which is widely used in the image retrieval community.

Training data. We construct two training datasets, namely Co-

cosyn and Landmark-clean-half. CocoSyn is a synthetic dataset

of 20,000 images. As described in Section 3.6, we synthesize 4

images for each segmented object instance. Together with the origi-

nal 4,000 Coco [19] images, we obtain 20,000 training images.

Landmark-clean-half is a subset of Landmark-clean dataset created

by Gordo et al. [10, 11]. The original Landmark-clean dataset used

for training DIR [10, 11] consists of 49,000 images with approximate
We use the released R-MAC model [11] to initialize the proposed local similarity. As shown in Table 3, the proposed local similarity trained on CocoSyn dataset and Landmark-clean-half dataset, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Oxf5k</th>
<th>Par6k</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiaMac [32]</td>
<td>77.0</td>
<td>84.1</td>
</tr>
<tr>
<td>∗(syn)</td>
<td>75.6</td>
<td>81.2</td>
</tr>
<tr>
<td>∗(syn+lch)</td>
<td>86.4</td>
<td>88.1</td>
</tr>
<tr>
<td>DIR-R-MAC [11]</td>
<td>83.9</td>
<td>93.8</td>
</tr>
<tr>
<td>∗(syn)</td>
<td>81.1</td>
<td>89.3</td>
</tr>
<tr>
<td>∗(syn+lch)</td>
<td>90.3</td>
<td>94.4</td>
</tr>
</tbody>
</table>

We evaluate the effectiveness of training with synthetic data using the source code and models released by the authors. It is slightly different from the performance reported in the paper (84.1 on Oxf5k and 93.6 on Par6k).

### 4.2 Effectiveness of Training with Synthetic Data

We use VGG-16 [40] and ResNet-101 [15] as our backbones. For models trained with synthetic data only, we use the model pre-trained on ImageNet as a starting point. For models that share feature extraction subnetwork with DIR-R-MAC [11], we use their released R-MAC model [11] to initialize them. We train the networks with stochastic gradient descent, with a learning rate of $10^{-3}$, momentum 0.9, weight decay $10^{-2}$ and batch size of 16. In each batch, we use 8 pairs of images (i.e., query images and their corresponding positive gallery images) to generate 8 positive pairs and 56 negative pairs. We select all 8 positive pairs and 8 most difficult negative pairs for loss calculation. Our implementation is based on PyTorch [24] library and trained on a PC with 4 NVIDIA GTX 1080Ti cards.

### 4.3 Effectiveness of Local Similarity

We use the released R-MAC model [11] to initialize the proposed feature extraction network. We further fine-tune the last 10 layers of the feature extraction subnetwork and the whole object localization subnetwork. As shown in Table 4, the proposed local similarity consistently outperforms global similarity [11] on all datasets. Moreover, combining local and global similarities with a linear weight of 0.9 for the local similarity has the best performance. This shows that our local similarity is complementary to the global similarity.

### 4.4 Impact of Scaling Up

#### Impact of prefiltering with global similarity

We do cross validation for linear combination weight $w$ from 0 to 1 and the size $K$ of the short list from 0 to 5,000. The results are shown in Fig. 7 and 8. We find that the performance is stable with a wide range of both parameters. The combined similarity is defined as $S = (1 - w) \times S_{global} + w \times S_{local}$, where $S_{global}$ and $S_{local}$ are the
Table 5: Comparing the performance of using different numbers of subvectors for compressing local features. * denotes the results after fine-tuning the object localization network with quantization. Full denotes the results without any memory and storage compression. CR denotes the compression ratio.

<table>
<thead>
<tr>
<th>M</th>
<th>32</th>
<th>32∗</th>
<th>64</th>
<th>64∗</th>
<th>128</th>
<th>128∗</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>256</td>
<td>256</td>
<td>128</td>
<td>128</td>
<td>64</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Oxf5k</td>
<td>84.9</td>
<td>87.3</td>
<td>88.1</td>
<td>89.0</td>
<td>90.4</td>
<td>90.7</td>
<td>91.6</td>
</tr>
<tr>
<td>Par6k</td>
<td>93.2</td>
<td>94.2</td>
<td>94.3</td>
<td>94.8</td>
<td>94.8</td>
<td>95.1</td>
<td>95.3</td>
</tr>
</tbody>
</table>

ResNet-101 model released by Gordo et al. [11] and a single scale as input. All implementations are carefully reproduced using the public source code released by the original authors. As shown in the first part of Table 6, our method trained with only the Landmark-clean-half outperforms all the other methods (compared without any post-processing). It can achieve a further gain when training with both synthetic data and real data. Again, with query expansion, our method performs best on all the datasets.

Table 6: Comparison with state-of-the-art methods. Our methods are marked with *. lch denotes the model trained with Landmark-clean-half only. lch+syn denotes the model trained with both Landmark-clean-half and CocoSyn. QE denotes query expansion.

<table>
<thead>
<tr>
<th>Without post-processing</th>
<th>Oxf5k</th>
<th>Par5k</th>
<th>Oxf105k</th>
<th>Par106k</th>
<th>OxfShf</th>
<th>ParShf</th>
<th>Ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>siaMAC [32]</td>
<td>77.7</td>
<td>84.1</td>
<td>70.1</td>
<td>76.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DIR-RMAC [11]</td>
<td>83.9</td>
<td>93.8</td>
<td>80.8</td>
<td>89.9</td>
<td>74.7</td>
<td>81.5</td>
<td>62.6</td>
</tr>
<tr>
<td>R-Match [11, 34]</td>
<td>88.1</td>
<td>94.9</td>
<td>85.7</td>
<td>91.3</td>
<td>83.5</td>
<td>86.9</td>
<td>71.0</td>
</tr>
<tr>
<td>DELF [23]</td>
<td>83.8</td>
<td>85.0</td>
<td>82.6</td>
<td>81.7</td>
<td>83.9</td>
<td>84.2</td>
<td>-</td>
</tr>
<tr>
<td>(lch)</td>
<td>90.5</td>
<td>95.7</td>
<td>88.6</td>
<td>92.5</td>
<td>90.4</td>
<td>93.3</td>
<td>71.1</td>
</tr>
<tr>
<td>(syn-lch)</td>
<td>90.8</td>
<td>95.7</td>
<td>88.9</td>
<td>93.0</td>
<td>90.5</td>
<td>92.7</td>
<td>76.5</td>
</tr>
<tr>
<td>With query expansion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>siaMAC+QE [32]</td>
<td>82.9</td>
<td>85.6</td>
<td>77.9</td>
<td>78.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DIR-RMAC+QE [11]</td>
<td>89.6</td>
<td>95.3</td>
<td>88.3</td>
<td>92.7</td>
<td>75.2</td>
<td>82.1</td>
<td>70.5</td>
</tr>
<tr>
<td>R-Match+QE [11, 34]</td>
<td>91.0</td>
<td>95.5</td>
<td>89.6</td>
<td>92.5</td>
<td>84.9</td>
<td>86.8</td>
<td>77.1</td>
</tr>
<tr>
<td>DELF+DIR+QE [11, 23]</td>
<td>90.0</td>
<td>95.7</td>
<td>88.5</td>
<td>92.8</td>
<td>84.4</td>
<td>84.6</td>
<td>-</td>
</tr>
<tr>
<td>(lch)</td>
<td>91.5</td>
<td>95.8</td>
<td>90.0</td>
<td>92.8</td>
<td>90.9</td>
<td>92.4</td>
<td>75.2</td>
</tr>
<tr>
<td>(syn-lch)</td>
<td>91.9</td>
<td>95.8</td>
<td>90.4</td>
<td>93.3</td>
<td>91.3</td>
<td>91.7</td>
<td>78.2</td>
</tr>
</tbody>
</table>

5 CONCLUSIONS

We proposed an end-to-end trainable CNN for local similarity learning by modeling the problem of object retrieval as object localization. Our CNN consists of a feature extraction subnetwork and an object localization subnetwork. Correlation layer is used to capture the spatial relations in images. We found that the correlation volume encodes whether the spatial relations of the gallery and those of the query are consistent or not. Thanks to the spatial relation harvesting, the proposed local similarity has excellent retrieval performance, and is complementary to global similarity. Besides, we proposed a scalable retrieval algorithm, by utilizing product quantization to compress gallery features, and global similarity to prefilter the gallery images and enhance the search results. Extensive experiments on challenging benchmarks demonstrate the effectiveness of the proposed algorithm.
REFERENCES


