Knowledge-Based Systems 186 (2019) 104955

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Exploring spatial and channel contribution for object based image retrieval☆,☆☆

Xiaoxia Shi [a,b], Xueming Qian [c,d,e,*,1]

[a] School of Software Engineering, Xi'an Jiaotong University, Xi'an, 710049, China
[b] SMILES Laboratory, Xi'an Jiaotong University, Xi'an, 710049, China
[c] Key Laboratory for Intelligent Networks and Network Security, Ministry of Education, Xi'an Jiaotong University, Xi'an, 710049, China
[d] SMILES Laboratory, School of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an, 710049, China
[e] Zhibian Technology Co., Ltd., Taizhou, 317000, China

ARTICLE INFO

Article history:
Received 12 March 2019
Received in revised form 12 August 2019
Accepted 15 August 2019
Available online 17 August 2019

Keywords:
Object retrieval
Spatial and channel contribution
Aggregate
Global representation vector

ABSTRACT

With the rapid development of deep learning methods, researchers have gradually shifted the research focus from hand-crafted features to deep features in the field of the content-based image retrieval (CBIR). A great deal of attention has been paid to aggregate the extracted features from the convolutional layer in the deep convolutional neural network (CNN) into a global representation vector for CBIR. In this paper, we propose a simple but effective method which called Strong-Response-Stack-Contribution (SRSC) to generate the global representation vector for object retrieval. As we know, for object retrieval, when using CNN to extract features, what we want is to extract features in the region of interest (ROI). So we explored spatial and channel contribution to help us focus more on ROI and make the global image representation vector more representative. The process of the approach SRSC is to first generate spatial contribution according to the degree of channel response intensity. Then, we generate channel contribution by joining the sparsity information and the element-value information together. Finally, the global representation vector is generated according to spatial and channel contribution to perform image retrieval. Experiments on Oxford and Paris buildings datasets show the effectiveness of the proposed approach.

1. Introduction

In the Web2.0 era, especially with the popularity of social networking sites such as Flickr and Facebook, young people are increasingly inclined to take photos with their mobile phones and share them online, which makes a huge amount of data on the Internet. In order to retrieve these massive image data more effectively, the content-based image retrieval (CBIR) technology has gradually become the main direction of image retrieval. The basic process is to extract the features of the low level (color, texture, etc.) from the image and then based on these features, CBIR measures similarity between the query image and dataset images, and finally getting the retrieval result.

However, there are several challenge problems in CBIR. One is the "semantic gap" problem [1]. In order to narrow the gap, one way is to design a good feature expression. In recent years,

CNNs have shown unprecedented advantages in image retrieval, target recognition and other fields. Researchers have chosen deep features [2–4] instead of hand-crafted local features to do image retrieval.

Object retrieval is an important part of CBIR. It consists on identify an object contained in the query image from the image database. The user is interested in the specific object in the image, so the result retrieved should be the images containing the object. For object retrieval, when using CNN to extract features, what we want is to extract features in region of interest (ROI). Based on such a way of thinking that, we propose a method to distinguish the object from the background.

☆ One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.104955.

☆☆ This work was supported in part by the NSFC, China under Grant 61732008, Grant 61772407, and Grant 1531141; in part by the National Key R&D Program of China under Grant 2017YFF0107700; and in part by the World-Class Universities (Disciplines), China and the Characteristic Development Guidance Funds for the Central Universities, China (PY3A022).

* Corresponding author at: SMILES Laboratory, School of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an, 710049, China.
E-mail addresses: sxx513@stu.xjtu.edu.cn (X. Shi), qianxm@mail.xjtu.edu.cn (X. Qian).
[1] Member, IEEE.

https://doi.org/10.1016/j.knosys.2019.104955
0950-7051/

In this paper, we propose a simple but effective method which called Strong-Response-Stack-Contribution (SRSC) to generate the global image representation vector for image retrieval. We explore two contributions that make us more focused on ROI. Our approach is inspired by CroW [5]. CroW proposed two weighting schemes, spatial and channel weight, which are very exciting. At the same time, we think about whether these two weighting schemes are optimal.

We design the spatial contribution based on following considerations: first, we think there is no need to add all the channel values to construct the spatial weight in CroW [5] because a lot of redundancy and noise are added. Second, PWA [6] proposed a channel selection strategy, which applies each selected channel as a weight to the features, and finally obtained the global representation vector with higher dimensions. We believe that the channel selection strategy can be used to select the appropriate channels instead of all channels. For channel contribution, we join the sparsity information and the element-value information together. In CroW, it only considers the sparsity information of the feature maps to construct the channel weight. However, a feature map is composed of real values, and cannot be considered as a binarized matrix. The element-value on each spatial location denotes the intensity information. So the channel weight strategy in CroW is not make full use of information. So we propose a new strategy to generate channel contribution using both information. We have improved the two weighting strategies proposed in CroW to make them work better.

The major contributions of this paper can be summarized as follows:

(1) We use the channel selection strategy [6] to select the strong response channels. Then we use the selected channels to build weight maps to generate the spatial contribution for assigning larger weights to the object location. It can help us distinguish the object from the background.

(2) We design a new channel weighting strategy called channel contribution for alleviating visual burstiness [7]. It makes full use of both sparsity and element-value information.

In the following sections, the related works are described in Section 2; the description of our method is in Section 3, 4, and 5; experiments, results and discussions are shown in Sections 6 and 7; Section 8 is the conclusion.

2. Related works

Content-based image retrieval task (CBIR) has long been an important research topic in the field of computer vision. Since the early 1990s, researchers have designed global features, local features, and convolutional features to explore the CBIR task, and achieving remarkable results. State-of-the-art methods are mainly based on two types of features, one for hand-crafted features and the other for deep convolutional features. The following subsections describe the existing works related to the above two aspects.

2.1. Methods based on hand-crafted features

In the field of image retrieval, a lot of research progress has been made in the image feature representation. In the early studies, the global features (color and texture) were used to match the images. For object retrieval, the influence of the shooting environment, such as the variations of illumination, scale, viewing angle and background, will have a greater impact on the search results. Due to the large environmental interference, for the same object image retrieval, when selecting features, it is often preferred to select those invariant local features with better anti-interference. So more detailed hand-crafted feature descriptors emerge gradually. One of the most representative is the classic scale-invariant feature transform (SIFT) [8–10] feature. Then diverse variants were proposed based on the SIFT features, such as PCA-SIFT [11], SURF [12], RootSIFT [13], etc. During the period from 2003 to 2012, image retrieval methods are mainly based on the SIFT features. Researchers mainly studied how to obtain the global feature representation vectors with a better representation ability by coding and aggregating local features. The proposed SIFT feature makes the Bag of Words (BoW) [14,15] model feasible. BoW method needs to generate a vocabulary book. A large-scale vocabulary book may contain 1 million or more visual vocabulary. So the approximation methods are crucial in assigning data to a large number of clusters. There are two representative works, they are Approximate k-means [16] and Hierarchical k-means [17]. In Approximate k-means, K clustering centers are indexed by random K–D trees [18]. Hierarchical k-means uses the standard k-means method for feature training in different levels. Because the size of the vocabulary book, the BoW histogram dimension is quite large. It gives a considerable computational complexity to the retrieval process, resulting in low retrieval speed and high memory consumption [19]. In 2008, Jégou et al. [20] proposed Hamming embedding, and medium vocabulary book began to occupy a place in image retrieval. In 2010, Perronnin et al. and Jégou et al. [21,22] proposed Fisher vectors (FV) [21] and Vector of Locally Aggregated Descriptors (VLAD) [22,23], and then small vocabulary book enters the researcher's horizon. FV is to use GMM to model feature points. GMM is actually a kind of clustering. It considers the distance from feature points to each cluster center. FV uses a linear combination of all cluster centers to represent the feature points. VLAD considers only the cluster center closest to the feature point, while preserving the distance of each feature point to the nearest cluster center. Besides these classic works, there are other methods to generate global representation vectors, like triangulation embedding [24], and etc [25–28].

2.2. Methods based on deep convolutional features

Recently, researchers confirm that the semantic information obtained by using CNN to extract image features is better than hand-crafted features. Features extracted from the convolutional layer or fully connected layer can be aggregated into global image representation vectors for object retrieval. Cosine distance or Euclidean distance are used to measure the similarity of images to complete the work of object retrieval.

The feature of the fully-connected layer provides a description of the high level of the image content, so early works [3,4,29] take the output of the last fully-connected layer directly as the global representation vector of the image. But some researchers [30,31] point out that the selection of the upper level layer is actually not conducive to the object retrieval, because of the features extracted from the upper layer losing the spatial information for the object. Then, more and more researches tend to use the features of the convolutional layer for object retrieval. When using the feature maps of the convolutional layer as the raw features for image retrieval, one of the main problems is how to convert the 3d tensor into a valid feature representation vector. Researchers have proposed a variety of solutions for this problem, like Sum-pooling, Average-pooling, Max-pooling and etc [5,32–34]. Sum-pooling sums all the pixel values of each feature map. So each feature map gets a real value, and the N feature maps get a vector with a length of N. Average-pooling is the same as Sum-pooling, except that the pixel values are summed and divided by the size of the feature map. Max-pooling selects the maximum pixel value of each feature map. In addition to direct feature pooling, it is also useful to assign specific weights to feature maps

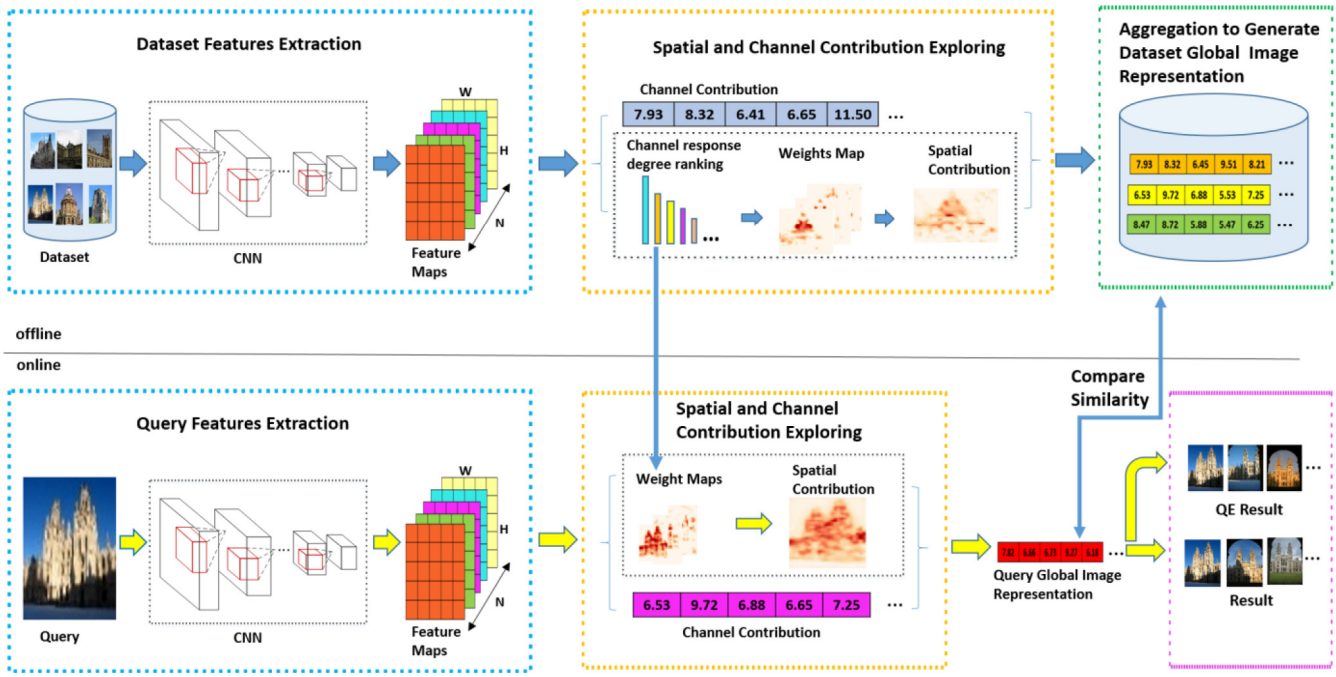

Fig. 1. The framework of our proposed method.

before pooling. Babenko et al. [32] apply a Gaussian mask to the feature maps before pooling. Kalantidis et al. [5] construct spatial and channel weights to increase the weights of ROI to a certain extent and reduce the weights of non-object regions. Wang et al. [35] designed an adaptive Gaussian filter and channel vector to co-weight the feature maps. In addition, there are some researches that perform regional analysis of convolutional features. Tolias et al. [33] proposed R-MAC. This method uses a specific region to perform max-pooling on the feature maps and generate a series of region vectors. Then it generates the image representation by using the Sum-pooling to aggregate the region vectors. One disadvantage of this approach is that it uses a fixed-position grid. In this regard, some methods have been improved on the basis of R-MAC. Jimenez et al. [34] improve the R-MAC using the (CAMs) approach. CAMs tries to combine the prediction information of the network category to make the spatial weight more discriminant. It generates a series of spatial maps representing the importance of each part of the image, which is related to the category information of the image. Similarly, Cao et al. [36] propose to obtain a series of basic regions directly from the convolutional layer and then use an adaptive rearrangement method.

3. System overview

Our system framework is shown in Fig. 1. As we can see, the whole work can be divided into two parts: offline and online. The offline part contains 5 steps, and the online part contains 6 steps.

The procedure of the offline system is described as follows: (1) Dataset features extraction. On the whole dataset, we extract the features of each image on the convolutional layer. (2) Channel response degree ranking. The extracted convolutional features generally contain hundreds of channels. For each dataset, we calculate the size of each channel's response and rank them in descending order. We select the top M channels. (3) Spatial contribution exploring. We use the top M channels to generate spatial contribution for each image. (4) Channel contribution exploring. We join the sparsity information and the element values information together to build the channel contribution vector. (5) Generate the global representation vector of each image on the whole dataset. We use the spatial contribution and channel contribution to aggregate features into the global image representation vector.

Online: (1) Query features extraction. We extract the features of the query image on the convolutional layer. (2) Spatial contribution exploring. We use the channel response degree information obtained in the offline part to generate spatial contribution. (3) Channel contribution exploring. (4) Generate the global representation vector of the query image. (5) Query expansion. Use simple query expansion to improve performance. (6) Similarity comparison. Compare the query global representation vector with datasets and get the results.

4. The proposed method on offline

In this section, we introduce the offline part of the system framework in detail. In Section 4.1, we introduce features extraction and present channel response degree ranking in Section 4.2. We present spatial contribution exploring in Section 4.3. Section 4.4 presents channel contribution exploring and Section 4.5 presents global image representation generating. Algorithm is given in Section 4.6.

4.1. Features extraction

In object retrieval, it is very important to find good features extraction and representation. As shown in Fig. 1, we input an image into CNN, and extract features on the convolutional layer. Let $X \in \mathcal{R}^{N\times W\times H}$ be the 3-dimensional feature tensor extracted from the selected convolutional layer, where N is the number of feature maps, H and W denote the height and width of the feature maps respectively.

In this paper, we use pre-trained VGG16 model to extract deep convolutional features. The network has 13 convolutional layers, 5 max-pooling layers and 3 fully connected layers. In our framework, we just extract deep features from the last pooling layer (the pool5 layer), and the extracted feature $X \in \mathcal{R}^{512\times W\times H}$.

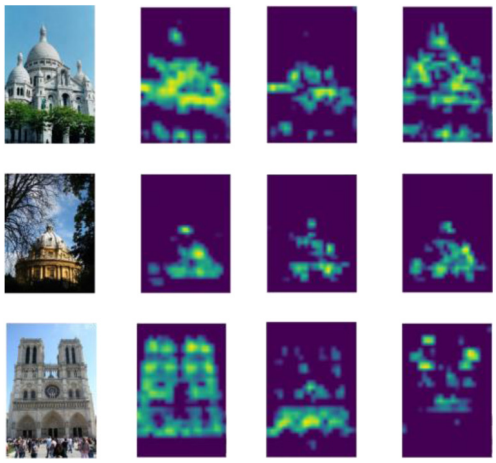

Fig. 2. Original images and visualization of the top 3 strong response channel values.

4.2. Channel response degree ranking

If a channel performs strong response on the whole dataset, this channel may be more discriminative. Therefore, we calculate the channel response on the entire dataset with J images. After that, we sort them from large to small, then select the top M $(M < N)$ normalized channel values to build weight maps.

We denote the entry in image j's feature X corresponding to channel n at spatial location (x, y) as $X_j(n, x, y)$. And we denote the entry in image j's h corresponding to channel n as $h\,(j, n)$. In order to calculate each channel's response, we first use Sum-pooling to aggregate the $N \times W \times H$ dimensional deep features X_j of image j $(j = 1, 2, \ldots, J)$, J is the total number of images in the dataset, and then get the vectors $h \in \mathcal{R}^{\mathrm{J}\times N}$ as follows:

$$h\,(j, n) = \sum_{x=1}^{W}\sum_{y=1}^{H} X_j\,(n, x, y) \quad \forall n \epsilon\,[1, 2, \ldots, N] \tag{1}$$

Then, we compute the average vector $\overline{h} \in \mathcal{R}^N$ of the h matrix as follows:

$$\overline{h}(n) = \frac{1}{\mathrm{J}}\sum_{j=1}^{\mathrm{J}} h\,(j, n) \tag{2}$$

Finally, we compute each channel's response $E \in \mathcal{R}^N$ as follows:

$$E(n) = \frac{1}{\mathrm{J}}\sum_{j=1}^{\mathrm{J}}(h\,(j, n) - \overline{h}(n))^2 \tag{3}$$

Let $V = \{v_1, v_2, \ldots, v_N\}, V \in \mathcal{R}^N$ stores the corresponding channel ID according to the size of the channel response E from large to small.

Fig. 2 shows the visualization of the top 3 strong response channel. We can see that each channel has a large response area, which may correspond to a part of an object or object, and obstructions such as trees and background are suppressed.

4.3. Spatial contribution exploring

we denote the entry in S at spatial location (x, y) as $S\,(x, y)$. And We denote the entry in $\mathcal{w}$ corresponding to channel m at spatial location (x, y) as $\mathcal{w}(m, x, y)$. According to the vector $V \in \mathcal{R}^N$, we select the top M channels values of feature maps $(M < N)$, and stack them into a weight maps $\mathcal{w} \in \mathcal{R}^{M\times W\times H}$. After adding multiple weight maps, those regions with large response are generally the areas where the objects are located. So we can use them to generate the spatial contribution of feature maps. The corresponding spatial contribution matrix $S \in \mathcal{R}^{(W\times H)}$ can be derived as follows:

$$S(x, y) = \sum_{m=1}^{M} \mathcal{w}(m, x, y) \tag{4}$$

The spatial contribution matrix S is then normalized and power-transformed. The specific operation is as follows:

$$S' = \left(\frac{S}{\sqrt{\sum S^2}}\right)^{\frac{1}{2}} \tag{5}$$

After doing this, we get final spatial contribution matrix $S' \in \mathcal{R}^{(W\times H)}$. Therefore, based on the spatial contributionS′, we generate intermediate vector $\Phi \in \mathcal{R}^N$ for channel contribution exploring by aggregating the spatial contribution matrix as follows:

$$\Phi(n) = \sum_{x=1}^{W}\sum_{y=1}^{H} X\,(n, x, y)\,S'(x, y) \tag{6}$$

Fig. 3 shows the visualization of the final spatial contribution and the top 15 strong response channel values. In Fig. 3, the leftmost column is the original image, the rightmost column is the visualized spatial contribution, and the middle part is the visualized top 15 strong response channels. The role of the spatial contribution is to assign larger weights to the features of the ROI. As can be seen from the middle column, the selected channels are the locations of the object or part of the object. We can add up these channels to assemble the whole object. So we accumulate these channels to get spatial contribution. As can be seen from the spatial contribution, the values of the object location are larger, and the values of the background such as trees and people are smaller. Therefore, the spatial contribution can reflect the ROI. Applying the spatial contribution as a weight to the image features can effectively detect ROI. Compared to CroW method that sums all channels' values, our method is a spatial contribution sensitive and which highlight objects from the whole redundant channels.

4.4. Channel contribution exploring

Our channel contribution strategy extends CroW by joining the sparsity information and the element-value information after doing spatial contribution of feature maps. In CroW, it only considers the sparsity of feature maps. It takes the number of non-zero elements as the information but it ignores the information contained in the specific values of feature maps.

Here, we propose a new method to generate channel contribution, which is based on the sparsity information and the element value information. We expect that similar images will have similar occurrence rates for a given feature. We also expect that similar images will have similar strong response for a given feature. We compute the proportion of non-zero responses $\mathbb{z}(n)$ and the element-value response $\mathbb{C}(n)$ for each channel n, and then compute the channel contribution $\mathcal{C} \in \mathcal{R}^N$:

$$\mathbb{z}(n) = \frac{\sum_{xy} 1[X(n, x, y) > 0]}{W \times H} \tag{7}$$

$$\mathbb{C}(n) = \left(\frac{\Phi(n)}{W \times H}\right)^2 \tag{8}$$

$$\mathcal{C}(n) = \alpha \times \log\left(\frac{N \times \varepsilon + \sum_d \mathbb{z}(d)}{\varepsilon + \mathbb{z}(n)}\right) + (1 - \alpha)$$

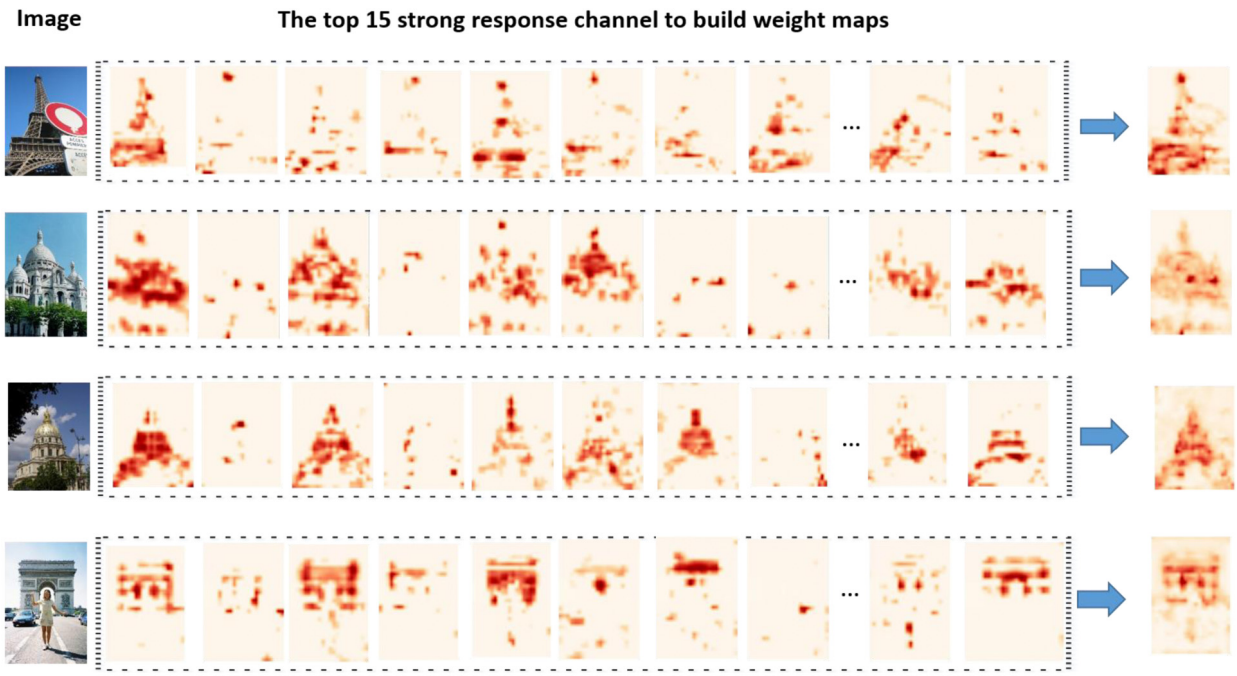

Fig. 3. Original images and visualization of final spatial contribution and the top 15 strong response channel.

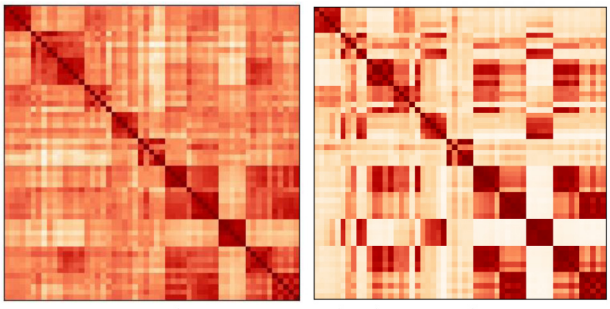

Fig. 4. The correlation of sparsity (a) and element-value response (b) for the query images of the Pari6K.

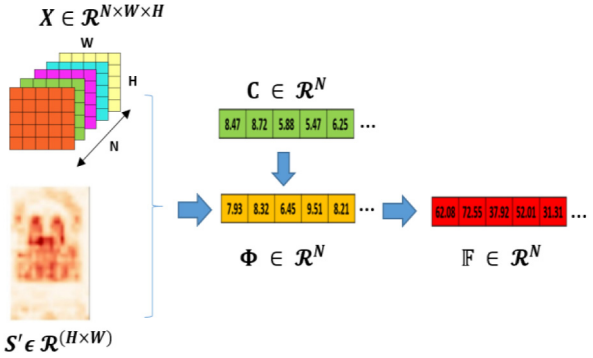

Fig. 5. The aggregation framework of our method.

$$\times \log\left(\frac{N \times \varepsilon + \sum_d \mathbb{C}(d)}{\varepsilon + \mathbb{C}(n)}\right) \tag{9}$$

where ε is a small constant added for numerical stability. We compute $\mathbb{C}(n)$ by summing the element value of each feature map after doing spatial contribution. α is a hyperparameter, in our experiments, we set $\alpha = 0.2$. We will discuss the parameter α in Section 7.2.

To explain why this channel contribution works, we visualize the pair-wise correlation of the vectors of channel sparsity $\mathbb{z}$ and element-value response $\mathbb{C}$ for all the query images of the Paris6K dataset. It has 55 query images, and each 5 images corresponds to a landmark of Paris. So all the query images can be classified into 11 classes. We organize images by class. As we can see from Fig. 4, both channel sparsity $\mathbb{z}$ and element-value response $\mathbb{C}$ are highly correlated for images of the same landmark. But element-value response $\mathbb{C}$ is less correlated for images of different landmarks. Therefore, compared with CroW [5] using sparsity information alone, we combine the sparsity and the element-value response information to generate the channel contribution can makes features more discriminative.

4.5. Global image representation generating

The final step in the offline part is to generate the global superior image representation. When we finish the exploring of spatial contribution and channel contribution, we can use them to aggregate local features into global image representation vector $\mathbb{F}$, $\mathbb{F} \in \mathcal{R}^N$ as follows:

$$\mathbb{F}(n) = \Phi(n) \odot \mathcal{C}(n) \tag{10}$$

where $\odot$ denotes element-wise product between vectors. The aggregation framework is shown in Fig. 5. As we can see, we applied spatial contribution $\boldsymbol{S}'$ to each channel of the feature maps X. Then we generate intermediate vector Φ by using Sum-pooling. Finally, we applied channel contribution $\mathcal{C}$ to Φ and get the global image representation vector $\mathbb{F}$.

4.6. Algorithm

Our algorithm aggregates the convolutional features into a global superior image representation. The pseudo code is in Algorithm 1. X is the 3-dimensional feature tensor extracted from the selected convolutional layer. It consists of N feature maps

with H height and W width ($X \in \mathcal{R}^{(N\times H\times W)}$). Let $V \in \mathcal{R}^N$ stores the corresponding channel ID according to the size of the channel response from large to small. $S' \in \mathcal{R}^{(H\times W)}$ denotes spatial contribution matrix and $\mathcal{C} \in \mathcal{R}^N$ denotes channel contribution vector.

Algorithm 1. The SRSC Framework for Aggregating Deep Convolutional Features

Input : 3d tensor $\boldsymbol{X}$, channel response degree ranking function $\boldsymbol{f_V}$, spatial contribution exploring function $\boldsymbol{f_s}$, channel contribution exploring $\boldsymbol{f_c}$, the parameters $\mathbf{M}$ and $\boldsymbol{\alpha}$

Output : $\boldsymbol{N}$-Dimensional global representation $\mathbb{F} \boldsymbol{\epsilon} \boldsymbol{\mathcal{R}^N}$

$\boldsymbol{V} = \boldsymbol{f_V}(\boldsymbol{X})$ // channel response degree ranking

$\boldsymbol{S} = \boldsymbol{f_s}(\boldsymbol{X}, \boldsymbol{V}, \boldsymbol{M})$ // Spatial contribution

$\boldsymbol{S'} = \boldsymbol{pnorm}(\boldsymbol{S})$ // Normalize and power scale

$\boldsymbol{\Phi}(\boldsymbol{n}) = \sum_{x=1}^{W}\sum_{y=1}^{H} \boldsymbol{X}(\boldsymbol{n}, \boldsymbol{x}, \boldsymbol{y})\, \boldsymbol{S'}(\boldsymbol{x}, \boldsymbol{y})$ // Intermediate vector

$\boldsymbol{\mathcal{C}} = \boldsymbol{f_c}(\boldsymbol{X}, \boldsymbol{\Phi}, \boldsymbol{\alpha})$ //Channel contribution

$\mathbb{F}(\boldsymbol{n}) = \boldsymbol{\Phi}(\boldsymbol{n}) \odot \boldsymbol{\mathcal{C}}(\boldsymbol{n})$ //Aggregate

5. The proposed method on online

In this section, we introduce the online part of the system framework in detail. Query features extraction, spatial contribution exploring, channel contribution exploring and generate the global representation vector of the query image are the same as the offline part, so this section will not go into details. It should be noted that the online part does not need to perform channel response degree ranking. We directly use the results generated by the offline part to perform spatial exploring. In Section 5.1, we introduce query expansion and present similarity comparison in Section 4.2.

5.1. Query expansion

We can improve performance through query expansion [37]. $\mathbb{F}(\mathrm{p})$ denotes the global representation vector of the image p. The process is to input a query image to get the global representation vectors of the top P images. Each vector is N-dimensional. We sum the P N-dimensional vectors to get the vector $\mathcal{Q} \in \mathcal{R}^N$, then L2 normalize $\mathcal{Q}$ to get the re-query vector $\mathcal{Q}' \in \mathcal{R}^N$. Finally, we can use this re-query vector to retrieval again to get the final results.

$$\mathcal{Q} = \sum_{p=1}^{P} \mathbb{F}(\mathrm{p}) \tag{11}$$

$$\mathcal{Q}' = \frac{\mathcal{Q}}{\sqrt{\sum_n \mathcal{Q}_n^2}} \tag{12}$$

Although it is simple, we show that it can improve the performance.

5.2. Similarity comparison

In content-based image retrieval, it is necessary to compare the similarity between the query image and the data set. In this paper, the final global image representation is a form of vector. We compare the similarities by comparing the query global image representation with the Euclidean distance of the database global image representations.

6. Experiments

In order to demonstrate the effectiveness of our Strong-Response-Stack-Contribution (SRSC) approach, we conduct experiments on the four benchmark datasets. In our approach, we set the parameter M = 15 and α = 0.2.Section 7.1 will discuss the selection of the parameter M and discuss the parameter α in Section 7.2. We set P = 3 and discuss this parameter in Section 7.3. All the features are extracted from the pool5 layer of the VGG16 [38] pre-trained in ImageNet without fine-tuning using Caffe. So the biggest dimension of the global representation vector is 512. We conduct PCA and whitening to reduce the dimension to 128 and 256. In this process, we used the whitening parameters learned on Paris6K when testing on Oxford5K (105K), and used the whitening parameters learned on Oxford5K when testing on Paris6K (106K). We use the cropped queries in Oxford5K (105K) and Paris6K (106K).

6.1. Performance evaluation

To make fair comparisons for state-of-the-art methods, we employ the standard protocol with other methods. We use mean average precision (mAP) to evaluate over all queries. It is the mean of the average precision scores for each query.

$$\mathrm{mAP} = \frac{\sum_{i=1}^{I} AP(i)}{I} \tag{13}$$

$$\mathrm{AP} = \frac{\sum_{k=1}^{K} (Pre\,(k) \times r\,(k))}{number\ of\ relevant\ images} \tag{14}$$

$$\mathrm{Pre} = \frac{number\ of\ relevant\ images}{number\ of\ retrieved\ images} \tag{15}$$

where I represents the total number of queries, K represents the number of retrieved images. $r(k)$ is an indicator function equaling 1 if the item at rank k is a relevant image, zero otherwise. We used the evaluation code provided on the public datasets to evaluate performance.

6.2. Datasets

Our method was experimented on four public datasets. Oxford5K [39] contains 11 different Oxford "landmarks" of images, each of them is retrieved from Flickr. The entire data set consists of 5062 high resolution (1024 × 768) images. For each landmark, the official provides 5 different queries to average the retrieval performance for any single query. Therefore, the official provides a total of 55 query images, these images are fixed, and the groundtruth of the 55 query images is officially provided. Paris6K [40] consists of 6412 images collected from Flickr by searching for particular Paris landmarks. It gives a set of 55 queries including 11 landmarks. Oxford105K and Paris106K are the extension of Oxford5K [39] and Paris6K [40] by adding additional 100K distracted images. The entire 100K dataset consist of high resolution (1024 × 768) images. These images were crawled from Flickr. Compared with the two small datasets of Oxford5K and Paris6K, this dataset contains more content and more noise. For example, the Oxford5K dataset is obtained from Flickr using tags such as "Oxford Christ Church" and "Oxford Radcliffe Camera"., while the 100K dataset contains coarse-grained tags such as "Friend" and "Holiday". Therefore, the two small datasets of Oxford5K and Paris6K are more limited and more representative.

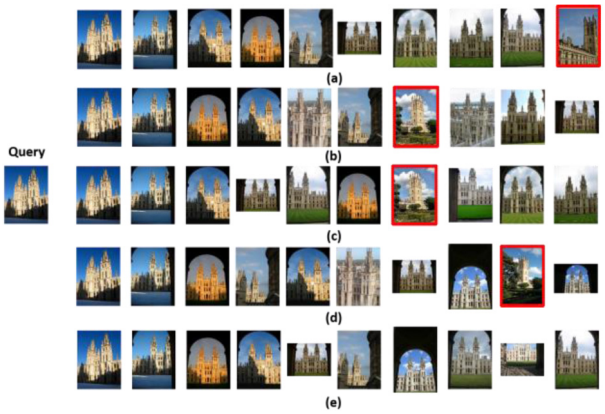

Fig. 6. Top 10 retrieval results for query "all_souls_000013". (a) Wang. (b) PWA. (c) R-MAC. (d) CroW. (e) SRSC.

Fig. 8. Top 10 retrieval results for query "cornmarket_000131". (a) Wang. (b) PWA. (c) R-MAC. (d) CroW. (e) SRSC.

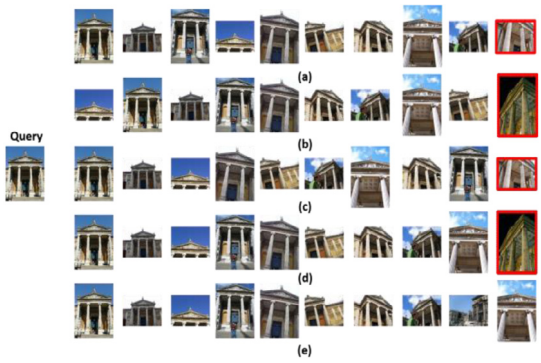

Fig. 7. Top 10 retrieval results for query "ashmolean_000007". (a) Wang. (b) PWA. (c) R-MAC. (d) CroW. (e) SRSC.

6.3. Comparison with the state-of-the-art

We compare our approach SRSC with CroW [5], NetVLAD [41], PWA [6], Neural code [3], R-MAC [33] and Wang et al. [35] in 3 different dimensions. Our main competitors are CroW [5], PWA [6] and Wang et al. [35]. In addition to the SRSC experiments compared with other methods, we also made experiments of SRSC with Query Expansion. Table 1 shows the experimental results of the state-of-the-art and our SRSC method using pre-trained VGG16 models on four benchmark datasets. The main competitors have been blackened in Table 1. As shown in table, we compare them in different dimensions. On the Paris6k dataset, when the final global image representation vectors are 128, 256 and 512 dimensions, the mAP of our method reaches 82.5%, 84.5% and 85.9% respectively. And on the Oxford5k dataset, the mAP of our method reaches 68.8%, 72.3% and 74.6% respectively. Our method SRSC can achieve the best performance in the comparison methods on the Oxford5K, Paris6K and Paris106K datasets. On the Oxford105K, when the global image representation vector is 128 dimensions, the performance is just a little lower than Wang [35]. After doing simple query expansion, the performance of our approach has risen in each dimension. The best performance on the two datasets (Paris6k and Oxford5K) is that mAP reaches 88.4% and 79.6% respectively.

Figs. 6–8 show the top 10 results for 3 example queries ("all_souls_000013", "ashmolean_000007" and "cornmarket_000131") for five methods (Wang [35], PWA [6], R-MAC [33], CroW [5] and SRSC) respectively. Unrelated results we identify with a red border. We used 512-dimensional final global image representation without query expansion. From Figs. 6 and 7, we can see that SRSC has the best performance and no errors in top 10. From Fig. 8, we can see that Wang [35] has the same performance as SRSC in top 5. There are three errors in top 10, but SRSC is relatively backward. In general, the performance of SRSC is quite good.

6.4. Experiments on Oxford105K and Paris106K

Considering that Oxford105K and Paris106K are large data sets, there are a lot of noise inside, so it is not suitable to select the corresponding channel based only on the channel response degree calculated over the entire dataset. We also calculated the channel response intensity of each image, and selected the intersection of top 30 and the top 30 of $V = \{v_1, v_2, \ldots, v_N\}$ to build the weight maps.

7. Discussion

In this section, we completely discuss the impact of different parameters. We also clearly state what it can bring for improvement, in terms of spatial contribution and channel contribution. Finally, we discuss the time complexity, memory consumption and the limitation of our method.

7.1. Discussion about the selection of the parameter M

When we generate spatial contribution, we choose the values of the top M strong response channels to build the weight maps. Experimental performance is related to the choice of M. In Section 4.2, we calculate each channel's response on the entire dataset and sort them from large to small. We believe that the top M channels already contain almost all the information when the top M channels' responses occupies most of the total responses of all channels on the entire dataset. For example, on the Oxford5k and Paris6K datasets, we set the threshold to 0.7, that is, the total responses of the top M channels occupy a ratio of 0.7 on the total responses of all channels. The M value calculated under this condition is 10 and 14 respectively. And we selected a series of values (M from 5 to 512) to conduct experiments on Paris6K and Oxford5K datasets. The choice of M and the final experimental result are shown in Fig. 9. As we can see, the experimental performance is considerable when M's choice is 10 and 14 respectively on the Oxford5K and Paris6K. And the experimental performance is also considerable when M is between 10 and 25, then the experimental performance decreases with the increase of the value of M. This also proves that the CroW

Table 1
Performance comparison with the state-of-the-art and our SRSC method with the different dimensions on the four benchmark datasets.

Method	Dim	Oxford5K	Paris6K	Oxford105K	Paris106K
NetVLAD [41]	128	61.4	69.5	-	-
Neural code [3]	128	55.7	–	52.3	–
CroW [5]	128	64.1	74.6	59.0	67.0
PWA [6]	128	64.5	76.9	–	–
Wang [35]	128	65.8	77.9	**61.4**	70.5
Our method (SRSC)	128	**68.8**	**82.5**	**60.6**	**71.8**
Neural code [3]	256	55.7	-	52.4	–
NetVLAD [41]	256	63.5	73.5	–	–
R-MAC [33]	256	56.1	72.9	47.0	60.1
CroW [5]	256	68.4	76.5	63.7	69.1
PWA [6]	256	68.7	79.6	–	–
Wang [35]	256	70.7	80.5	66.5	74.0
Our method (SRSC)	256	**72.3**	**84.5**	**66.8**	**74.6**
NetVLAD [41]	512	67.6	74.9	-	-
Neural code [3]	512	55.7	–	52.2	–
R-MAC [33]	512	66.9	83.0	61.6	75.7
CroW [5]	512	70.8	79.7	65.3	72.2
PWA [6]	512	72.0	82.3	66.2	75.8
Wang [35]	512	72.8	83.0	68.1	76.3
Our method (SRSC)	512	**74.6**	**85.9**	**69.0**	**77.1**
CroW+QE	128	67.0	79.3	64.1	72.8
CroW+QE	256	71.8	81.5	67.6	75.3
CroW+QE	512	74.9	84.8	70.6	79.4
PWA+QE	512	74.8	86.0	72.6	80.7
Wang+QE	128	68.3	82.3	68.4	70.6
Wang+QE	256	73.8	85.1	73.4	78.2
Wang+QE	512	77.0	87.4	74.2	80.9
SRSC+QE	128	**74.3**	**85.0**	**68.4**	**74.8**
SRSC+QE	256	**77.0**	**87.2**	**73.5**	**79.4**
SRSC+QE	512	**79.6**	**88.4**	**74.8**	**81.6**

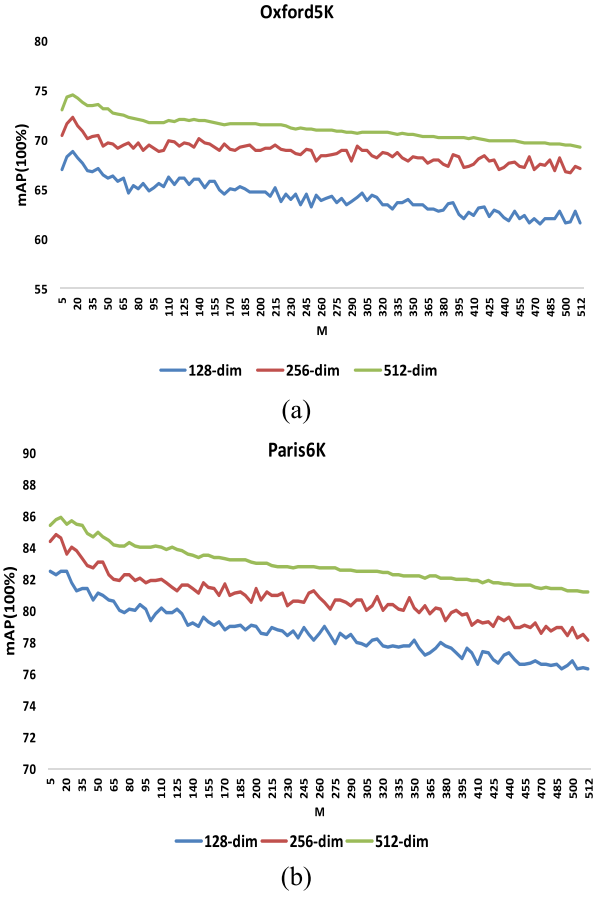

Fig. 9. Performance with different values of M when tested on (a) Oxford5K and (b) Paris6K.

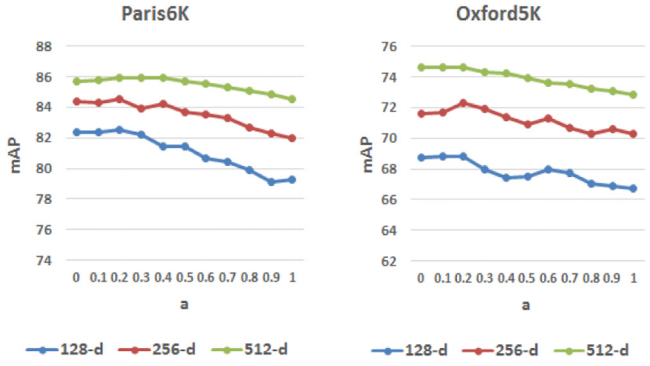

Fig. 10. Performance with different values of α when tested on Paris6K and Oxford5K.

method cannot achieve the best experimental performance by adding all the channel values to construct spatial weights because a lot of redundancy and noise are added. The performance (mAP) of our method on Paris6K and Oxford5K datasets is higher than that of CroW method by 6%∼8% and 4% respectively.

7.2. Discussion about the parameter α in Eq. (9)

α is a hyperparameter, which determines the contribution of the sparsity information and the element-value information to generate channel contribution. We set the value of $\alpha = [0.1, 0.2, \ldots, 0.9, 1.0]$. We performed experiments on Oxford5K and Paris6K respectively, and performed experiments on three dimensions (128, 256, 512) on each dataset. The experimental results are shown in Fig. 10. From the figure we can see that when α is 0.2, the experimental performance is the best. This also shows that the element-value information has a greater impact on channel contribution.

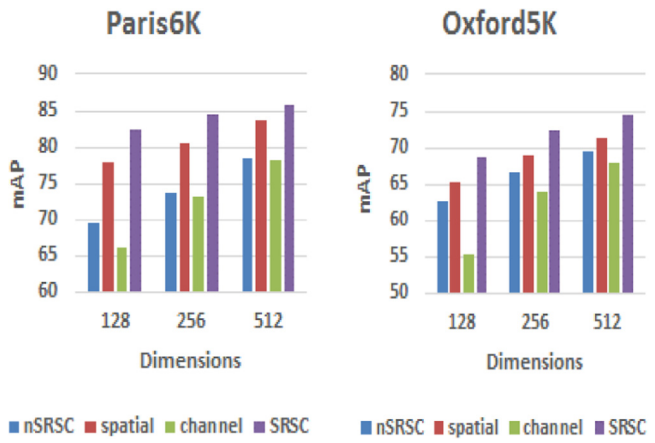

Fig. 11. Performance of four scenarios. (1) nSRSC (2) spatial (3) channel (4) SRSC.

Table 2
Comparisons of performance under the different value of parameter p.

P	0	1	2	3	4	5
mAP	68.8	71.2	73.0	74.3	73.7	73.6

Dimension = 128

P	0	1	2	3	4	5
mAP	72.3	74.5	76.4	77.0	77.0	76.6

Dimension = 256

P	0	1	2	3	4	5
mAP	74.6	76.1	78.3	79.6	78.9	79.0

Dimension = 512

7.3. Discussion about the parameter P in query expansion

For the selection of P, it was done a set of experiments on Oxford5K. We set $P = [0, 1, 2, 3, 4, 5]$, and the results are shown in Table 2. Experiments have shown that $P = 3$ can achieve the best performance. As for the computational cost, it is almost the same. Selecting the larger value or the smaller value, the program running time differs by only 0.1–1 s, which is almost negligible because it is just a simple addition of top P features.

7.4. Discussion about the spatial and channel contribution

To explore the impact of spatial contribution and channel contribution on performance, we chose four scenarios to experiment: (1) nSRSC, not use SRSC, just sum pooling, (2) spatial, spatial contribution alone, (3) channel, channel contribution alone, and (4) SRSC. The experimental results are shown in Fig. 11. From the figure we can see three points: (1) Using spatial contribution alone can improve the experimental performance to some extent. (2) Using channel contribution alone will reduce the performance. (3) Combining the use of spatial contribution and channel contribution can optimize experimental performance. This result is in line with our expectations because our approach is inspired by TF–IDF, a numerical statistic that relates the Term Frequency to Inverse Document Frequency. The spatial contribution is similar to TF, and the channel contribution is similar to IDF. When we do not multiply the TF and use IDF alone, it will be counterproductive.

7.5. Discussion about the time complexity

The time complexity of the algorithm is analyzed theoretically and tested in practice.

Table 3
The running time of different stages.

Stage	Time (s)
Features extraction (CPU)	48.03
Features extraction (GPU)	0.23
Channel response intensity calculation (CPU)	0.01
The global representation vector generation	0.02
Similarity comparison	1.03

The complexity of the offline part is mainly divided into three parts: features extraction, channel response intensity calculation and generation of the global representative vector. So the offline time complexity is about $J \times (O(\sum_{l=1}^{D} B_l^2 \times T_l^2 \times U_{l-1} \times U_l) + O(N \times W \times H) + O(N \times W \times H))$, Where D represents the number of convolutional layers of the neural network, l represents the lth convolutional layer of the neural network, U_l represents the number of output channels of the lth convolutional layer of the neural network, B denotes the edge length of the output feature map, and T represents the side length of each convolution kernel. The extracted feature consists of N feature maps with H height and W width. J represents the number of the dataset's images. We can see that the main time consumed in offline part is the extraction of convolutional features. So the offline time complexity can be described as $J \times (O(\sum_{l=1}^{D} B_l^2 \times T_l^2 \times U_{l-1} \times U_l))$.

The complexity of online part can be divided into features extraction of query, the global representation vector generation and similarity comparison. So the online time complexity is about $O(\sum_{l=1}^{D} B_l^2 \times T_l^2 \times U_{l-1} \times U_l) + O(N \times W \times H) + O(N \times J)$. The main time consumed in online part is also the extraction of convolutional features. So the online time complexity can be described as $O(\sum_{l=1}^{D} B_l^2 \times T_l^2 \times U_{l-1} \times U_l)$

If other methods also use convolutional features for retrieval, then the speed should not be too different, unless other methods use accelerated retrieval.

We also actually tested the running time. The running time of each stage is shown in Table 3. The test environment is Ubuntu 18.04, the deep learning framework is Caffe, the programming language is python3.6. The CPU model is Intel Xeon X5687, and the GPU model is GeForce GTX 1070. We use 55 query images to test and use the average time of each stage. In the feature extraction stage, when using CPU, it takes 48.03 s for each image, and 0.23 s for GPU. The offline part also has channel response intensity calculation. We calculated the channel response intensity on the Oxford5K dataset, it takes 0.01 s to run on the CPU. The global vector generation and similarity comparison are all performed on the CPU, and the required time is 0.02 s and 1.03 s, respectively.

7.6. Discussion about the memory consumption

In our method, we use VGG16 network to extract features, our aggregation strategy determines that the longest dimension of our global representation vector is 512. We can also use the dimensionality reduction strategy to reduce the global vector to 256 or even 128 dimensions, and the performance is still considerable. On Oxford5k, the memory required is the space to store the 5062*512 matrix. Compared to the BoW method using the SIFT features, the storage space is greatly reduced. Because BoW needs to generate 1 million or more visual vocabularies, each SIFT feature is 128-dimensional. On Oxford5k, it needs 5062*128 million space.

7.7. Discussion about the limitation

Our approach is unsupervised. We extract features using a pre-trained model on ImageNet. This model is trained from more than

1.28 million images and contains 1000 categories. We input an image into this model to extract the convolutional features as the deep features of the image, and then apply our proposed method to the features to obtain the global representation vector. So the limitation of our method is that if the image is very different from the images content on ImageNet, such as medical images and remote sensing images, the extracted features may not be able to express the images effectively. For such images, the model can be fine-tuned using the images of these types.

8. Conclusion

In this paper, we propose a method of aggregating convolutional features into a global representation vector for image retrieval. The key characteristic of our method is that we designed spatial contribution and channel contribution to help us focus on ROI. Spatial contribution can help us distinguish objects from the background. Channel contribution can alleviate visual burstiness [7]. It makes full use of both sparsity and element-value information. Experiments on four benchmark datasets demonstrate that our approach is better than the state-of-the-art aggregation methods. Our method does not require thousands of vector dimensions [6,42,43] to represent an image. The maximum dimension depends on how many channels there are in the features extraction layer of neural networks. The experimental performance of reducing it to a lower dimension by dimension reduction is also considerable. The simple global representation vector also makes the memory burden less heavy.

References

[1] R. Hong, Y. Yang, M. Wang, et al., Learning visual semantic relationships for efficient visual retrieval, IEEE Trans. Big Data 1 (4) (2015) 152–161.
[2] H. Azizpour, A. Sharif Razavian, J. Sullivan, et al., From generic to specific deep representations for visual recognition, in: CVPR Workshops, 2015, pp. 36–45.
[3] A. Babenko, A. Slesarev, A. Chigorin, et al., Neural codes for image retrieval, in: ECCV, vol. 1, 2014, pp. 584–599.
[4] A. Sharif Razavian, H. Azizpour, J. Sullivan, et al., CNN features off-the-shelf: an astounding baseline for recognition, in: CVPR Workshops 2014, pp. 512–519.
[5] Y. Kalantidis, C. Mellina, S. Osindero, Cross-dimensional weighting for aggregated deep convolutional features, in: ECCV Workshops, vol. 1, 2016, pp. 685–701.
[6] J. Xu, C. Shi, C. Qi, et al., Part-based weighting aggregation of deep convolutional features for image, in: AAAI, 2018, pp. 7436–7443.
[7] H. Jégou, M. Douze, C. Schmid, On the burstiness of visual elements, in: CVPR, 2009, pp. 1169–1176.z.
[8] D. Lowe, G, Distinctive image features from scale-invariant keypoint, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[9] D.G. Lowe, Local feature view clustering for 3d object recognition, in: CVPR, vol. 1, 2001, pp. 682–688.
[10] C. Kang, L. Zhu, X. Qian, et al., Geometry and topology preserving hashing for SIFT feature, IEEE Trans. Multimed. 21 (6) (2019) 1563–1576.
[11] Y. Ke, R. Sukthankar, Pca-sift: a more distinctive representation for local image descriptors, in: CVPR, vol. 2, 2004, pp. 506–513.
[12] H. Bay, T. Tuytelaars, L. Van Gool, Surf: speeded up robust features, in: ECCV, vol. 1, 2006, pp. 404–417.
[13] R. Arandjelović, A. Zisserman, Three things everyone should know to improve object retrieval, in: CVPR, 2012, pp. 2911–2918.
[14] J. Sivic, A. Zisserman, Video Google: A text retrieval approach to object matching in videos, in: ICCV 2003, pp. 1470–1477.
[15] G. Csurka, C. Dance, L. Fan, et al., Visual categorization with bags of keypoints, in: Workshop on statistical learning in computer vision, ECCV, 1, (1–22) 2004, pp. 1–2.
[16] J. Wang, J. Wang, Q. Ke, et al., Fast approximate k-means via cluster closures, in: Multimedia Data Mining and Analytics, 2015, pp. 373–395.
[17] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: CVPR, vol. 2, 2006, pp. 2161–2168.
[18] V. Lepetit, P. Lagger, P. Fua, Randomized trees for real-time keypoint recognition, in: CVPR, vol. 2, 2006, pp. 2161–2168.
[19] Z. Li, X. Zhang, H. Müller, et al., Large-scale retrieval for medical image analytics: A comprehensive review, Med. Image Anal. 43 (2018) 66–84.
[20] H. Jegou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, in: ECCV, vol. 1, 2008, pp. 304–317.
[21] F. Perronnin, Y. Liu, J. Sánchez, et al., Large-scale image retrieval with compressed fisher vectors, in: CVPR, 2010, pp. 3384–3391.
[22] H. Jégou, M. Douze, C. Schmid, et al., Aggregating local descriptors into a compact image representation, in: CVPR, 2010, pp. 3304–3311.
[23] R. Arandjelovic, A. Zisserman, All about VLAD, in: CVPR, 2013, pp. 1578–1585.
[24] H. Jégou, A. Zisserman, Triangulation embedding and democratic aggregation for image search, in: CVPR, 2014, pp. 3310–3317.
[25] N. Murray, H. Jégou, F. Perronnin, et al., Interferences in match kernels, IEEE Trans. Pattern Anal. Mach. Intell. 39 (9) (2017) 1797–1810.
[26] S. Pang, W. Zhang, L. Zhu, et al., Beyond Sum and Weighted Aggregation: An Efficient Mixed Aggregation Method with Multiple Weights for Image Search, in: ACM Multimedia (Thematic Workshops), 2017, pp. 59–67.
[27] X. Qian, Y. Zhao, J. Han, et al., Image location estimation by salient region matching, IEEE Trans. Image Process. 24 (11) (2015) 4348–4358.
[28] Y. Xue, X. Qian, B. Zhang, et al., Mobile image retrieval using multi-photos as query, in: ICME Workshops 2013, pp. 1-4.
[29] Y. Gong, L. Wang, R. Guo, et al., Multi-scale orderless pooling of deep convolutional activation features, in: ECCV, vol. 7, 2014, pp. 392–407.
[30] G. Tolias, R. Sicre, H. Jégou, Particular object retrieval with integral max-pooling of CNN activations, CoRR abs/151105879 (2015).
[31] J. Hao, J. Dong, W. Wang, et al., What is the best practice for cnns applied to visual instance retrieval? abs/161101640 (2016).
[32] A. Babenko, V. Lempitsky, Aggregating local deep features for image retrieva, in: ICCV 2015, pp. 1269–1277.
[33] G. Tolias, R. Sicre, H. Jégou, Particular object retrieval with integral max-pooling of CNN activations, CoRR abs/151105879 (2015).
[34] A. Jimenez, J.M. Alvarez, X. Giro-i Nieto, Class-weighted convolutional features for visual instance search, CoRR abs/170702581 (2017).
[35] J. Wang, J. Zhu, S. Pang, et al., Adaptive co-weighting deep convolutional features for object retrieval, in: ICME, 2018, pp. 1–6.
[36] J. Cao, L. Liu, P. Wang, et al., Where to focus: Query adaptive matching for instance retrieval using convolutional feature maps, CoRR abs/160606811 (2016).
[37] M. Cimpoi, S. Maji, A. Vedaldi, Deep filter banks for texture recognition and segmentation, Int. J. Comput. Vis. 118 (1) (2016) 65–94.
[38] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/14091556 (2014).
[39] J. Philbin, O. Chum, M. Isard, et al., Object retrieval with large vocabularies and fast spatial matching, in: CVPR, 2007, pp. 1–8.
[40] J. Philbin, O. Chum, M. Isard, et al., Lost in quantization: Improving particular object retrieval in large scale image databases, in: CVPR, 2008, pp. 1–8.
[41] R. Arandjelovic, P. Gronat, A. Torii, et al., NetVLAD: CNN architecture for weakly supervised place recognition, IEEE Trans. Pattern Anal. Mach. Intell. 40 (6) (2018) 1437–1451.
[42] A. Sharif Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, in: CVPR Workshops, 2014, pp. 806–813.
[43] J. Donahue, Y. Jia, O. Vinyals, et al., Decaf: A deep convolutional activation feature for generic visual recognition, CoRR abs/13101531 (2013).