Aggregated Deep Feature from Activation Clusters for Particular Object Retrieval

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

Zhanghui Kuang
Sense Time
Hong Kong
kuangzhanghui@sensetime.com

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

Wei Zhang
Sense Time
Hong Kong
wayne.zhang@sensetime.com

ABSTRACT

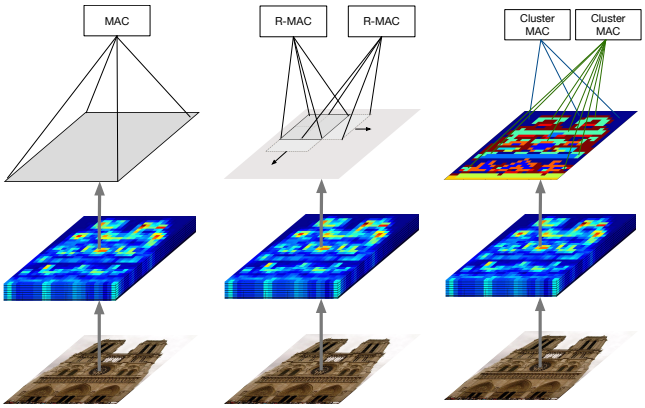
This paper introduces a clustering based deep feature for particular object retrieval. Many object retrieval algorithms focus on aggregating local features into compact image representations. Recently proposed algorithms, such as R-MAC and its variants, aggregate maximum activations of convolutions from rectangular regions of multiple scales and have achieved state-of-the-art performance. Such rectangular regions, however, cannot fit the "non-rectangular" shape of an arbitrary object well, and therefore cover much clutter in the background. This paper targets at mitigating this problem by proposing a deep feature based on clustering the activations of convolutions and aggregating the maximum activations from such clusters. Compared with the square regions used in R-MAC, the clusters thus obtained can better fit the arbitrary shapes and sizes of the objects of interest. By not taking spatial location into account, it is possible to have a single cluster covering multiple disconnected regions that correspond to repeated but isolated visual patterns. This helps to avoid over-weighting such patterns in the aggregated feature. Experiments are carried out on the challenging Oxford5k and Paris6k datasets, and results show that our clustering based deep feature outperforms the R-MAC feature.

KEYWORDS

Particular object retrieval; activation clustering

1 INTRODUCTION

Particular object retrieval has attracted sustained research attention for decades. It aims at retrieving all images containing the same instance of a certain object given in a query image. Traditional methods [6, 7, 23, 24, 27, 37] extract local visual features such as SIFT [22], and then aggregate them into sparse or compact feature vectors which can be searched efficiently. Since CNNs [20] have achieved great successes in fundamental vision tasks such as image classification [15, 19, 36, 38], image detection [9, 10, 33] and image segmentation [5, 14, 21], a number of CNN based methods [3, 4, 11–13, 29, 39] have been proposed for extracting discriminative deep activation features for image retrieval. Many of these methods [12, 13, 31, 32, 39] focus on aggregating local activations into compact image representations.



(a) MAC [32] (b) R-MAC [39] (c) Our method

Figure 1: Different strategies for max-pooling. (a) MAC [32] max-pools from the whole convolutional feature maps. (b) R-MAC [39] max-pools from multiple-scale square regions of the feature maps. (c) Our method max-pools from clusters of activations (represented by different colors).

Ideally, an optimal image representation for particular object retrieval should only encode the activations inside the regions of interest (ROI) (*i.e.*, regions corresponding to the objects of interest) and ignore those outside the ROIs (*i.e.*, those corresponding to the background). However, ROIs depend not only on the image itself but also on a user's intent, and thus cannot be known in advance. Razavian *et al.* [32] did not consider ROIs and proposed to use maximum activations of the whole convolutional layers (MAC) as an image representation. The information of the ROIs might be suppressed in MAC due to global max-pooling. Later, Tolias *et al.* [39]

*The work originated from Zhenfang's summer internship at Sense Time.

ThematicWorkshops'17, October 23–27, 2017, Mountain View, CA, USA

ACM ISBN 978-1-4503-5416-5/17/10…$15.00
https://doi.org/10.1145/3126686.3126696

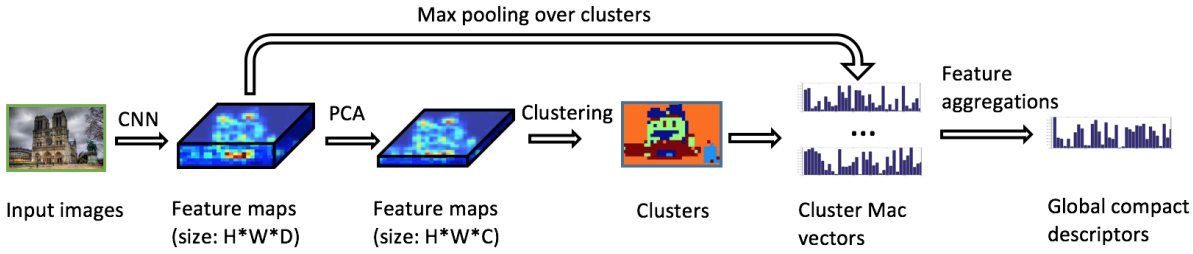


Figure 2: The whole framework of the proposed method

proposed to aggregate regional maximum activations of convolutional layers (R-MAC). Although aggregating activations that are max-pooled from local regions may be more robust than MAC, the square regions used cannot fit the "non-rectangular" shape of an arbitrary object well, and therefore often cover much clutter in the background. Rather than max-pooling from square regions, Gordo *et al.* [12, 13] adopted the region proposal network (RPN) [33] to produce ROIs for max-pooling. However, its performance depends heavily on the training data, and might not generalize well to test images from a very different source.

In this paper, we propose a clustering based deep feature for particular object retrieval. Activations are first partitioned into clusters, and are then max-pooled from each of the clusters. A compact image representation is then obtained by aggregating these maximum activations from the clusters. In contrast to the previous approaches [12, 13, 31, 32, 39] in which activations are max-pooled from rectangular regions, activations are max-pooled here from clusters with arbitrary shapes and sizes (see Figure 1c). As regions of similar visual patterns often generate similar activations, our approach tends to perform max-pooling from regions of similar visual patterns, and is therefore less likely to be influenced by clutter in the background. By not taking spatial location into account, it is also possible to generate a single cluster covering multiple disconnected regions that correspond to repeated but isolated visual patterns. This helps to avoid over-weighting such patterns in the aggregated feature. Experiments are carried out on the challenging Oxford5k [27] and Paris6k [28] datasets, and results show that our clustering based feature outperforms the R-MAC feature.

The rest of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 details our proposed method. Section 4 summarizes our experimental results, with concluding remarks in Section 5.

2 RELATED WORK

Existing methods in the literature for particular object retrieval can roughly be categorized into two approaches, namely traditional local invariant feature based approach and deep feature based approach.

Traditional local invariant feature based approach. Sivic and Zisserman [37] first proposed the Bag-of-Word (BoW) model which represents a set of local visual features (*e.g.*, SIFT [22]) in an image as a sparse feature vector, and then searching can be carried out efficiently by inverted index. Since then, large visual codebooks [2, 23], spatial verification [24, 27], and query expansion [6, 7] were proposed to improve its performance in terms of efficiency and accuracy. To reduce the memory cost, Fisher Vector [25, 26] and VLAD [1, 16] were proposed to aggregate local features into compact representations.

Deep feature based approaches. Babenko *et al.* [4] and Razavian *et al.* [30] are the pioneers in investigating deep features in fully-connected layers for image retrieval. Razavian *et al.* [31] extensively studied the availability of image representations based on the convolutional network. Babenko and Lempitsky [41] found that sum-pooling over convolutional layers with a centering prior can produce promising performance. Kalantidis *et al.* [18] proposed spatial and feature channel weighting that significantly improves the performance. Azizpour *et al.* [32] introduced a highly competitive compact image representation by a global max-pooling operation. Later, Tolias *et al.* [39] proposed max-pooling of activations from multiple-scale square regions, and aggregated the maximum activations into a compact feature vector. Our method is inspired by [39], but different in that we perform max-pooling from clusters of activations. Very recently, Radenovic *et al.* [29] learned compact representations in an end-to-end fashion using a pairwise loss. Their training strategy is complementary to our proposed method, and can further improve our performance. Similarly, Gordo *et al.* [12, 13] learned representations using a ranking loss. Besides, they employed RPN [33] to produce ROIs for max-pooling. Different from [12, 13], our clusters for max-pooling are generated in an unsupervised manner, non-rectangular in shape, and do not depend on the training data.

3 CLUSTERING BASED DEEP FEATURE

This section introduces our clustering based deep feature for particular object retrieval. The framework for generating our aggregated deep feature is similar to the one described in [39], with the major difference in the way we carry out the max-pooling operation.

An overview of our framework is illustrated in Figure 2. We first reduce the dimension of the CNN features using PCA, and employ Normalized Cut [35] to cluster these PCA features. We then compute per-cluster MAC vectors by max-pooling from the clusters, and carry out l_2 normalization and PCA whitening on these per-cluster vectors. Finally, we aggregate these per-cluster feature vectors and l_2-normalize the resulting vector to produce our final deep feature.

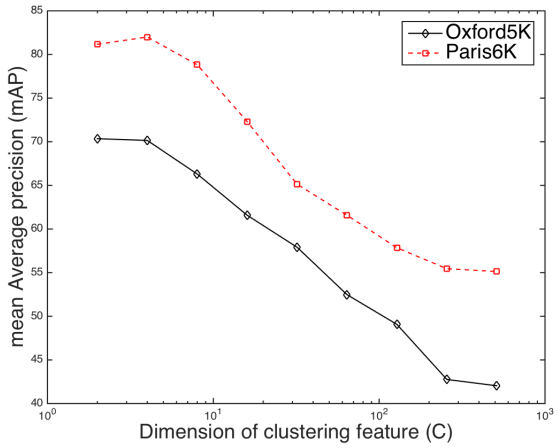


Figure 3: Image retrieval performance using feature clusters generated by l_2 norm distances of C-dimensional PCA features. The retrieval results are conducted on the conv5_3 layer of the VGG16 model [36] pre-trained on ImageNet classification tasks.

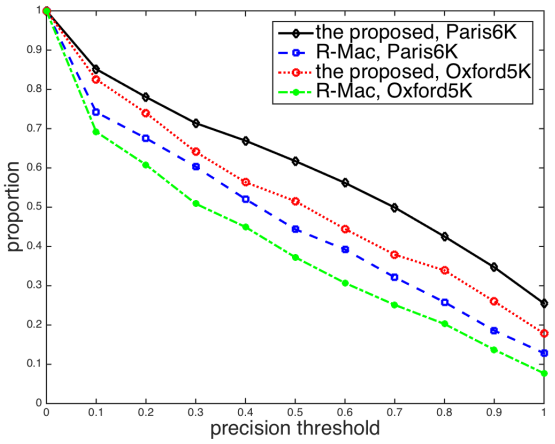


Figure 4: The proportion of regions that meet a particular precision threshold. For R-MAC[39], we extract approximately 20 regions per image at 3 different scales, which is the same as the original paper. For the proposed method, we hierarchically cluster the features into 5 and 15 clusters respectively, which also generate a total of 20 regions per image.

3.1 PCA dimension reduction

The activation features generated by the convolutional layer can be considered as a set of D-dimensional descriptors extracted at $H \times W$ spatial locations. Before clustering activation features, we use PCA to reduce the dimension of these descriptors from D to C (where $D \gg C$). There are two main reasons for carrying out this step. First, we find empirically that increasing the dimension of the descriptors often does not improve the retrieval performance but instead makes it worse (see Figure 3). Note that, in principle, most of the valuable information is encoded in the first few dimensions of a PCA feature, and the last few dimensions of the feature can be seen as noise. Figure 5 provides a visualization of the content in different dimensions of the PCA feature. We can see that the difference between object and background is large in the first few dimensions, and the difference decreases rapidly as the dimension increases. Second, it is easy to see that a small value of C makes it fast and efficient to cluster the features, and again the clustering will also be less affected by noise.

3.2 Feature clustering

We observe that activation features of object regions have larger l_2 norm values when compared with those of the background (see examples in the second row of Figure 7). This suggests that it is possible to segment the object by simply applying some unsupervised clustering methods on the feature vectors. We apply Normalized Cut [35] to produce the clustering results. Figure 7 shows some examples of the clustering results using Normalized Cut.

From Figure 7, we can see the benefits of using feature clusters for computing the MAC features. First, although different background objects such as sky, cloud and trees may have quite different appearances in the color space, they do have similar features in the PCA-dimension-reduced feature space and can often be clustered into the same cluster. Since the final descriptor is generated by aggregating the per-cluster MAC vectors, having all background objects in the same cluster would mean they have less weight in the final descriptor. Second, object regions often have very distinguishable feature vectors that makes most of the resulting clusters overlap with these regions. This implies objects of interest generally have more weight in the final descriptor. Third, a cluster can have arbitrary shape and size, and can cover multiple spatially disconnected regions. This allows clusters to better fit the arbitrary shape and geometry of the objects, and produce features that can better describe them.

To better illustrate the benefits of using feature clusters, we define the following precision measure

$$precision = \frac{area(rgn) \cap area(gt)}{area(rgn)}, \tag{1}$$

where $area(rgn)$ represents the area of a particular region of interest and $area(gt)$ is the area of the bounding box of the query object. We test on the query set of Oxford5k [27] and Paris6k [28] datasets, which provide the ground truth bounding boxes of the query objects. We calculate the percentage of the regions having a precision larger than or equal to a particular threshold. The results are shown in Figure 4, from which we can see that the regions generated by feature clustering are much more precise than the square regions used in R-MAC[39].

Since objects in different images may have different sizes, we further propose to cluster the feature map hierarchically to minimize the effect of object size. This can be done efficiently by simply setting different numbers of target clusters. Besides, in order to get more diverse and robust clusters, we also resize the images to

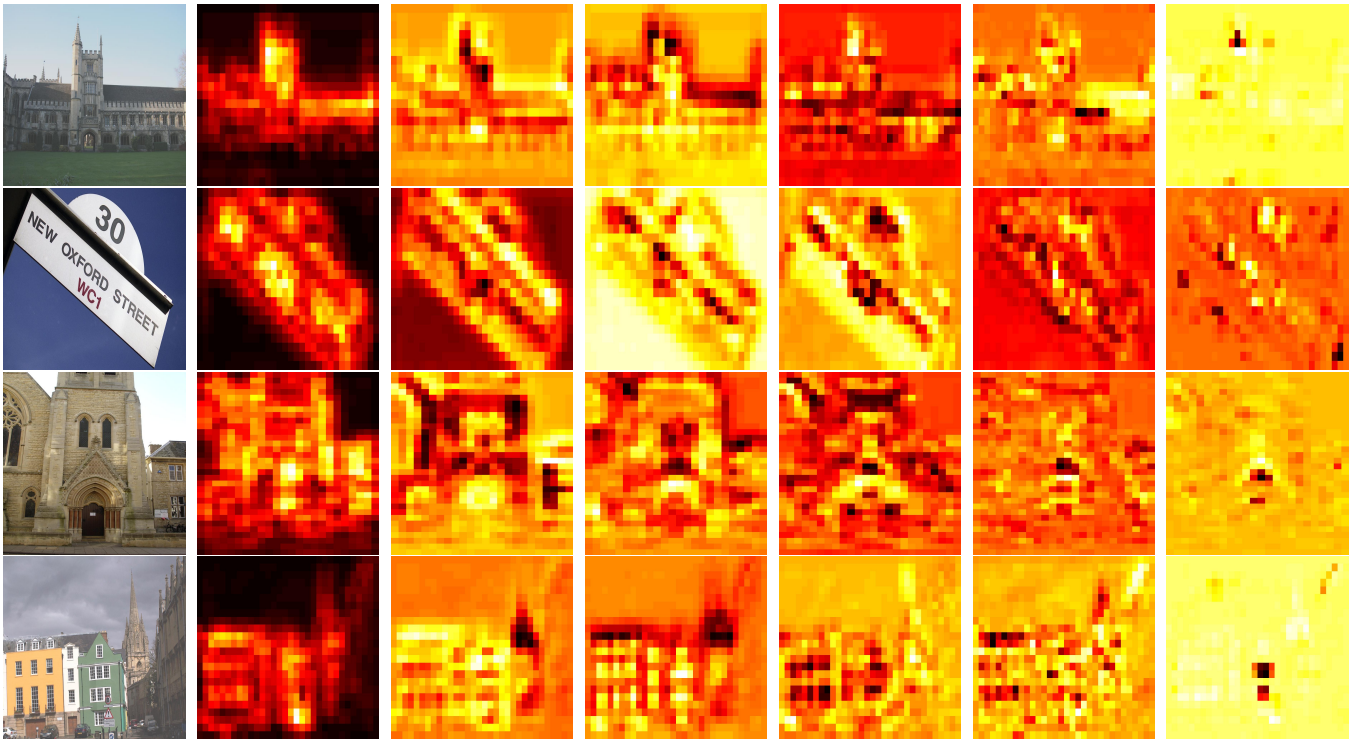


Figure 5: A visualization of the content in different dimensions of a PCA feature. The first column is the input images, and the 2-7 columns are a visualization of the distribution of values for the 1st, 2nd, 8th, 64th, 256th, 512th dimension, respectively, of the conv5_3 descriptor after PCA. The difference in color represents the difference in value in a particular dimension. Images are sampled from the Oxford5k dataset [27], and the CNN used is the VGG16 model [36] pre-trained on ImageNet classification tasks [34].

the same spatial dimension as the feature map and apply Normalized Cut [35] to the color space of the resulting image to obtain additional clusters.

Normalized cut [35] performs clustering based on the similarity matrix of samples. We calculate the feature similarity matrix based on the weighted l_2 distance between C-dimensional PCA features. The distance between the i-th feature and the j-th feature is defined as

$$d_{i,j} = \sqrt{\sum_{k=1}^{C} w_k (f_{i,k} - f_{j,k})^2} \tag{2}$$

where $f_{i,k}$ denotes the value of the k-th dimension of the i-th feature, $w_k = \frac{\sigma_k}{\sum_{m=1}^{C} \sigma_i}$ and σ_m is the variance of the m-th dimension of the PCA feature. Supposed N is the total number of features, we construct a distance matrix $\mathbf{D} = (d_{i,j}) \in R^{N \times N}$ using (2). Based on this distance matrix, we define the similarity matrix between features as

$$\mathbf{S} = (e^{-(\frac{d_{i,j}}{z})^2}) \in R^{N \times N}, \tag{3}$$

where z is a scale. In our experiments, we set $z = 0.05 \times \max(d_{i,j})$.

Unlike the conventional image segmentation tasks in [8, 35], we on purposely do not take the spatial distance of the features into account when constructing the similarity matrix. Intuitively, features that are close to each other in the image space are deemed to be similar and should be more likely to be clustered into the same cluster. We do observe that constructing a similarity matrix based on both feature distance and spatial distance as in [35] enables spatial smoothness within the clusters. However, in our case, we find that adding spatial information actually degrades the retrieval performance. According to our observations, adding spatial information tends to over-segment large background regions (*e.g.*, sky and land). This results in over-weighting the corresponding features in the aggregated feature. Besides, adding spatial information will also prevent repeated but isolated visual patterns from being clustered into the same cluster. This again results in over-weighting the corresponding features in the aggregated feature. Figure 6 show the effect of spatial distance on feature clustering.

3.3 Comparison with previous aggregation methods

The activations from a convolutional layer can be represented by a 3D tensor χ of $H \times W \times D$ dimensions, where D denotes the number of channels in the convolutional layer. Such a 3D tensor can be viewed as a $H \times W$ feature map with each feature being described by a D-vector. A D-dimensional MAC vector [32] of an image can be generated by max-pooling over this feature map, *i.e.*,

$$\mathbf{f} = [f_1 ... f_k ... f_D]^{\mathrm{T}} \text{ with } f_k = \max_{p \in \Omega} \chi_k(p), \tag{4}$$

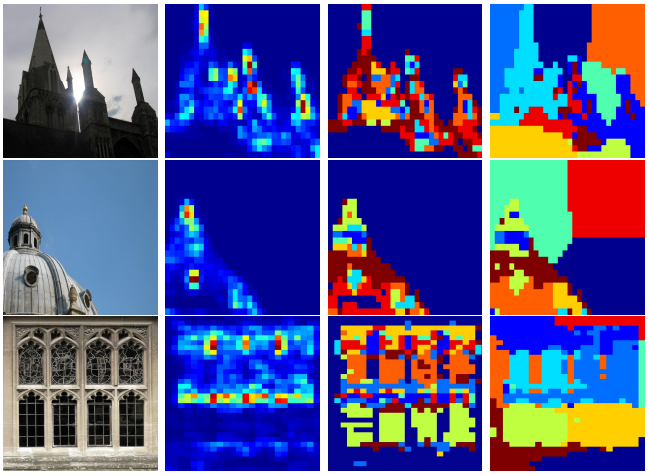


Figure 6: Effect of spatial distance on feature clustering. The first column shows the input images sampled from the Oxford5k [27] dataset. The second column shows l_2 norm of conv5_3 features extracted from the VGG16 model [36] pre-trained on ImageNet classification tasks [34]. The third column shows the clustering results based on feature distance only. The last column shows the clustering results using both feature distance and spatial distance. Different colors represent different clusters generated by Normalized Cut [35].

where $\chi_k(p)$ denotes the value of the k-th dimension of the feature vector at the spatial location p, and Ω represents all the valid spatial locations. It is easy to see that MAC is in fact an extreme case of our method with the cluster number set to one.

Different from MAC which max-pools over all the spatial locations, R-MAC [39] generates a compact image representation by computing regional MAC vectors and summing them up. Given a feature map from a convolutional layer, R-MAC first defines square regions on the feature map via a sliding window strategy. For each square region, a MAC vector is computed by max-pooling over all spatial locations within the region. Finally, a global descriptor is obtained by summing up all these regional MAC vectors. The R-MAC vectors of two images can be compared by computing the dot product between them, and this operation is equivalent to an approximate many-to-many region matching. Our method is similar to R-MAC in that it also generates a global descriptor by aggregating regional MAC vectors. Unlike R-MAC in which the regions are square in shape and are defined independent of the image content, our method generates regions through feature clustering based on feature similarity. This makes our regions, and hence our regional MAC vectors, more content/object-aware.

Inspired by Faster-RCNN [33], Gordo *et al.* [12, 13] proposed to replace the rigid grid used to define regions in R-MAC with a region proposal network (RPN). The main idea behind the RPN is to predict a set of candidate boxes of various size and aspect ratios, depending on the image content. Similar to using RPN to generate regions, our method also relies on the convolutional features of the images in generating regions. However, different from [12, 13], our method is totally unsupervised and class-agnostic, requiring no training data with bounding box annotation, and the output regions do not need to be rectangular or connected.

4 EXPERIMENTS

In this section, we discuss the implementation details of our method, and compare our results against those produced by state-of-the-art methods.

4.1 Experimental details

We evaluate our method on the Oxford buildings [27] and Paris [28] datasets, which are composed of 5,063 and 6,412 images respectively. These two datasets are referred to as Oxford5k and Paris6k, respectively. Besides, as in [39], 100k Flickr images [27] are added to these two datasets to form the Oxford105k and Paris106k datasets for evaluation at a larger scale. We use the VGG16 [36] and ResNet101[15] as our CNN models as they are widely used in the literature. We use MatConvNet [40] or caffe [17] to extract the convolutional feature maps according to the format of the pre-trained models. Retrieval performance is measured in terms of mean average precision (mAP). We follow the standard protocol for Oxford5k and Paris6k, and crop the query images with the provided bounding boxes. In order to have a direct comparison with [29], we also evaluate queries generated by cropping the feature maps of the input images. To have a fair comparison, we always use input images with the same resolution as the ones used by the methods against which we are comparing. For the implementation of Normalized Cut, we use the source code released by Shi et al. [35]. For the VGG16 model, we use PCA to reduce the feature dimension to 4 before calculating the feature similarity matrix. For the ResNet101 model, we reduce the feature dimension to 7. The post-processing PCA whitening matrix is learned on Oxford5k when testing on Paris6k, and vice versa. Experiments are conducted in Matlab 2014b on a machine with an Intel(R) Core i7-4790 CPU processor(3.60GHz) and ubuntu operating system. For the generation of regions in our method, we use the features to generate 5 and 20 clusters respectively, and then we use RGB values of the resized images to produce 5 additional regions. The average time for clustering in our experiments is about 0.15 second with the unoptimized Matlab code.

4.2 Comparison with R-MAC [39]

To compare with R-MAC [39] fairly, we use the same experiment setting as R-MAC. For each region, we first calculate a MAC vector [3] and then post-process this MAC vector with l_2 normalization, PCA whitening and l_2 normalization. Finally, we sum the regional MAC vectors and l_2-normalize the resulting vector again to get a final descriptor. The results are shown in Table 1.

From Table 1, we can see that our method outperforms the original R-MAC [39] in most cases both in the original ranking and post re-ranking. Besides, we notice that the improvements in the Oxford datasets are larger than that in the Paris datasets. This can be explained by the fact that the images in the Oxford dataset have more clutter in the background than those in the Paris dataset. As discussed previously, the square regions used by R-MAC will cover much clutter in the background that may contaminate the regional MAC vectors. On the other hands, the clusters used in our method

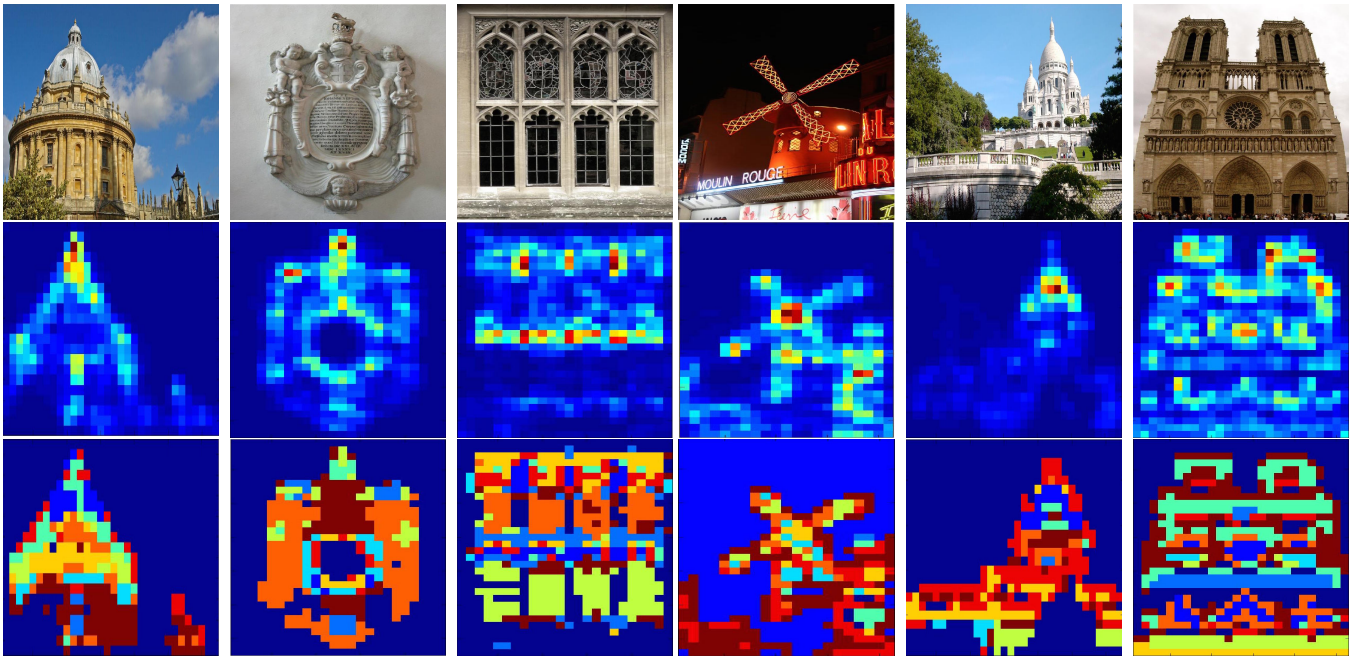


Figure 7: Clustering results on the Oxford5k [27] and Paris6k [28] datasets.The first row shows the input images. The second row shows the l_2 norm of the D-dimensional descriptors at each spatial location extracted from the VGG16 model [36] pre-trained on ImageNet classification tasks [34]. The last row shows the clustering results using Normalized Cut [35]. The number of clusters is set to 10, and different clusters are represented using different colors.

Table 1: Comparison with R-MAC [39]. *crop I* means using images cropped with the given bounding boxes as input; *crop X* means using the whole image as input and then cropping activations that fall inside the bounding boxes. *R* means Re-ranking; *QE* means query expansion.

Method	Dim.	datasets			
		Oxf5k	Par6k	Oxf105k	Par106k
R-MAC, *crop I*	512	66.9	83.0	61.6	75.7
Ours, *crop I*	512	**70.3**	**83.6**	**65.0**	**76.2**
R-MAC, *crop X*	512	70.3*	84.3*	65.1*	77.6*
Ours, *crop X*	512	**73.8**	83.9	**69.7**	76.4
R-MAC, *crop I* + *R* + *QE*	512	77.3	86.5	73.2	79.8
Ours, *crop I* + *R* + *QE*	512	**81.4**	**87.5**	**79.0**	**81.2**
R-MAC, *crop X* + *R* + *QE*	512	79.5*	86.7*	75.7*	81.1*
Ours, *crop X* + *R* + *QE*	512	**82.5**	**86.8**	**80.0**	80.5

∗ our implementation

can better fit the shapes and sizes of the objects of interest, and thus can produce better regional descriptors. We also observe that in most situations, using cropped activations as input produces a better performance than using cropped images as input. This suggests cropping activation may provide more useful contextual information rather than adding more distracting noises from the pixels outsides the bounding box of the query object.

Table 2: Comparison with siaMac [29]. *crop I* means using images cropped with the given bounding boxes as input; *crop X* means using the whole image as input and then cropping activations that fall inside the bounding boxes; *R* means Re-ranking; *QE* means query expansion; PCA_w means using the same PCA whitening as mentioned in R-MAC [39]; L_w means the learning whitening in siaMac [29].

Method	Dim.	datasets			
		Oxf5k	Par6k	Oxf105k	Par106k
siaMac, *crop I*	512	76.3	84.5	68.5	77.1
Ours, *crop I*	512	**80.9**	**86.7**	**76.4**	**80.0**
siaMac, *crop X* + L_w	512	80.1	85.0	74.1	77.9
Ours, *crop X* + PCA_w	512	**81.4**	**87.6**	**78.2**	**81.4**
siaMac, *crop X* + L_w + *R* + *QE*	512	84.5	86.4	80.4	79.7
Ours, *crop I* + PCA_w + *R* + *QE*	512	**85.5**	**88.9**	**84.2**	83.6
Ours, *crop X* + PCA_w + *R* + *QE*	512	**85.5**	**88.9**	83.9	**84.3**

4.3 Comparison with siaMac [29]

To further demonstrate the benefits of generating regions through feature clustering based on feature similarity, we compare the performance of our method against siaMac [29]. The post-processing and aggregation of regional vectors are the same as in the original R-MAC [39]. We use input images with the same resolution as the ones used in [29]. We do not apply a learned whitening to the regional vectors, and use the same PCA whitening method as described in R-MAC [39]. The results are shown in Table 2.

From Table 2, we can see that performance of our method is always better than that of using grid regions on VGG16 fine-tune

Table 3: Comparison with DIR [13]. *crop I* **means using images cropped with the given bounding boxes as input;** PCA_w **means using the same PCA whitening as mentioned in R-MAC [39].**

Method	Dim.	datasets	
		Oxf5k	Par6k
Ours, $crop\ I + PCA_w$	2048	82.7	93.8
DIR-grid [12]	2048	84.1	93.6
DIR-RPN [12]	2048	85.2	94.0

model [29], even when their results are obtained using the learned whitening L_w as described in [29] whereas ours are obtained using the same PCA whitening PCA_w as described [39], and that it has been proved in [29] that L_w is more effective than PCA_w .

4.4 Comparing with DIR[13]

We also test the performance of our method on the fine-tuned model provided by Gordo *et al.* [13]. One thing to note is that this fine-tuned model is trained end-to-end, with the post-processing PCA (shifting and fully connected layer) learned in the network. In our method, we first extract its Res5c_relu convolutional feature maps and then pools over the clusters. we post-process each cluster MAC vectors with standard l_2 normalization, PCA whitening and l_2 normalization again to get the final descriptors. The PCA matrix of our method is learned on Paris6k when testing on Oxford5k and vice versa, whereas the PCA (Shift and fully connected layer) of DIR is learned with the Landmarks-clean datasets [4, 13] end-to-end. The results are shown in Table 3.

Comparing with DIR-grid [12], we have slightly better performance on the Paris6k dataset but a little bit worse performance on the Oxford5k datasets. From experiments in the 4.2 and the 4.3, we have proved that our cluster-based method is better than those grid-based methods. Our explanation for why we have worse performance on oxford datasets is that the end-to-end learned PCA (shifting and fully connected layer) may have served as something like a metric learning technique, making the final regional vectors more suitable to be measured with cosine distance than the regional vectors after normal PCA whitening. Actually, we have carried out experiments, replacing the post-processing [12] of DIR-grid with our PCA whitening post-processing, its performance on Oxford5k and Paris6k datasets has dropped to 81.7 and 92.7 respectively. Although we want to fine-tune a shifting and fully connected layer for our method using the landmark-clean datasets [4, 13], we find that many URLs of the datasets has become invalid and some of the images has wrong annotations, which fails our training. Finally, despite the fact that using end-to-end PCA and RPN [33] can further improve the performance on the Paris and Oxford datasets, they need massive training data and the results are dependent on the training data. On the other hand, our method is totally unsupervised and can be applied to any pre-trained CNN model without any fine-tuning.

5 CONCLUSION

In this paper, we introduce an aggregated deep feature based on feature clustering for particular object retrieval. Our method is similar to R-MAC in that it generates a global descriptor by aggregating regional MAC vectors. Unlike R-MAC in which the regions are square in shape and are defined independent of the image content, our method generates regions through feature clustering based on feature similarity. This makes our regions, and hence our regional MAC vectors, more content/object-aware. Experimental results show that our method outperforms R-MAC. Compared with RPN based method, our method has the advantage of not requiring any training data with bounding box annotations, being class-agnostic, and capable of being applied to any pre-trained models without any fine-tuning.

REFERENCES

[1] Relja Arandjelovic and Andrew Zisserman. 2013. All About VLAD. In *CVPR*. 1578–1585.
[2] Yannis Avrithis and Yannis Kalantidis. 2012. Approximate gaussian mixtures for large scale vocabularies. In *ECCV*. 15–28.
[3] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. 2015. From generic to specific deep representations for visual recognition. In *CVPRW*. 36–45.
[4] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. 2014. Neural codes for image retrieval. In *ECCV*. 584–599.
[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2015. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*.
[6] Ondrej Chum, Andrej Mikulik, Michal Perdoch, and Jiri Matas. 2011. Total recall II: Query expansion revisited. In *CVPR*. 889–896.
[7] O Chum, James Philbin, and Josef Sivic. 2007. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*.
[8] Timothee Cour, Florence Benezit, and Jianbo Shi. 2005. Spectral segmentation with multiscale graph decomposition. In *CVPR*, Vol. 2. 1124–1131.
[9] Ross Girshick. 2015. Fast R-CNN. In *ICCV*.
[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*. 580–587.
[11] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*. 392–407.
[12] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. 2016. Deep image retrieval: Learning global representations for image search. In *ECCV*. 241–257.
[13] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. 2016. End-to-end learning of deep visual representations for image retrieval. *arXiv preprint arXiv:1610.07940* (2016).
[14] Kaiming He and Ross Girshick. 2017. Mask R-CNN. arXiv:1703.06870
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[16] H Jégou and Matthijs Douze. 2010. Aggregating local descriptors into a compact image representation. In *CVPR*.
[17] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*. 675–678.
[18] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. 2016. Cross-dimensional Weighting for Aggregated Deep Convolutional Features. In *ECCVW*. 42.
[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*. 3431–3440.
[22] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *IJCV* 60, 2 (2004), 91–110.
[23] David Nister and Henrik Stewenius. 2006. Scalable recognition with a vocabulary tree. In *CVPR*, Vol. 2. 2161–2168.
[24] Michal Perdoch, Ondrej Chum, and Jiri Matas. 2009. Efficient representation of local geometry for large scale object retrieval. In *CVPR*. 9–16.
[25] Florent Perronnin and Christopher Dance. 2007. Fisher kernels on visual vocabularies for image categorization. In *CVPR*. 1–8.
[26] Florent Perronnin, Yan Liu, Jorge Sánchez, and Herve Poirier. 2010. Large-scale image retrieval with compressed fisher vectors. In *CVPR*. 3384–3391.
[27] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*. 1–8.

[28] J Philbin, O Chum, M Isard, J Sivic, and A Zisserman. 2008. Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In *CVPR*.

[29] Filip Radenović, Giorgos Tolias, and Ondrej Chum. 2016. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *ECCV*. 3–20.

[30] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, Stefan Carlsson, Ali Sharif, Razavian Hossein, Azizpour Josephine, Sullivan Stefan, and K T H Royal. 2014. CNN Features off-the-shelf : an Astounding Baseline for Recognition. In *CVPRW*. 512–519.

[31] Ali Sharif Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. 2014. Visual instance retrieval with deep convolutional networks. *arXiv preprint arXiv:1412.6574* (2014).

[32] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. 2014. A Baseline for Visual Instance Retrieval with Deep Convolutional Networks. *CoRR* abs/1412.6574 (2014).

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*. 91–99.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and Others. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115, 3 (2015), 211–252.

[35] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *T-PAMI* 22, 8 (2000), 888–905.

[36] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

[37] Josef Sivic and Andrew Zisserman. 2003. Video Google: a text retrieval approach to object matching in videos. In *ICCV*, Vol. 2. 1470–1477.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Chapel Hill, and Ann Arbor. 2014. Going Deeper with Convolutions. In *CVPR*.

[39] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2016. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*.

[40] Andrea Vedaldi and Karel Lenc. 2015. Matconvnet: Convolutional neural networks for matlab. In *ACM MM*. 689–692.

[41] Artem Babenko Yandex and Victor Lempitsky. 2015. Aggregating local deep features for image retrieval. In *ICCV*.