Int J Comput Vis (2017) 124:237–254
DOI 10.1007/s11263-017-1016-8

End-to-End Learning of Deep Visual Representations for Image Retrieval

Albert Gordo[1] · **Jon Almazán**[1] · **Jerome Revaud**[1] · **Diane Larlus**[1]

Received: 19 December 2016 / Accepted: 5 May 2017 / Published online: 5 June 2017

Abstract While deep learning has become a key ingredient in the top performing methods for many computer vision tasks, it has failed so far to bring similar improvements to instance-level image retrieval. In this article, we argue that reasons for the underwhelming results of deep methods on image retrieval are threefold: (1) noisy training data, (2) inappropriate deep architecture, and (3) suboptimal training procedure. We address all three issues. First, we leverage a large-scale but noisy landmark dataset and develop an automatic cleaning method that produces a suitable training set for deep retrieval. Second, we build on the recent R-MAC descriptor, show that it can be interpreted as a deep and differentiable architecture, and present improvements to enhance it. Last, we train this network with a siamese architecture that combines three streams with a triplet loss. At the end of the training process, the proposed architecture produces a global image representation in a single forward pass that is well suited for image retrieval. Extensive experiments show that our approach significantly outperforms previous retrieval approaches, including state-of-the-art methods based on costly local descriptor indexing and spatial verification. On Oxford 5k, Paris 6k and Holidays, we respectively report 94.7, 96.6, and 94.8 mean average precision. Our representations can also be heavily compressed using product quantization with little loss in accuracy.

Keywords Deep learning · Instance-level retrieval · Visual search · Visual representation

Communicated by Svetlana Lazebnik, Cordelia Schmid.

✉ Diane Larlus
Diane.Larlus@xrce.xerox.com

Albert Gordo
Albert.Gordo@xrce.xerox.com

Jon Almazán
Jon.Almazan@xrce.xerox.com

Jerome Revaud
Jerome.Revaud@xrce.xerox.com

[1] Computer Vision Group, Xerox Research Center Europe, Meylan, France

1 Introduction

Instance-level image retrieval is a visual search task that aims at, given a query image, retrieving all images that contain the same object instance as the query within a potentially very large database of images. Image retrieval and other related visual search tasks have a wide range of applications, *e.g.*, reverse image search on the web or organization of personal photo collections. Image retrieval has also been seen as a crucial component for data-driven methods that use visual search to transfer annotations associated with the retrieved images to the query image (Torralba et al. 2008). This has proved useful for annotations as diverse as image-level tags (Makadia et al. 2008), GPS coordinates (Hays and Efros 2008), or prominent object location (Rodriguez-Serrano et al. 2015).

Deep learning, and particularly deep convolutional neural networks (CNN), have become an extremely powerful tool in computer vision. After Krizhevsky et al. (2012) achieved the first place on the ImageNet classification and localization challenges in 2012 (Russakovsky et al. 2015) using a convolutional neural network, deep learning-based methods have significantly improved the state of the art in other tasks such as object detection (Girshick et al. 2014) and semantic segmentation (Long et al. 2015). Recently, they have also shined in other semantic tasks such as image captioning (Frome et al. 2013; Karpathy et al. 2014) and visual question answering

(Antol et al. 2015). However, deep learning has been less successful so far in instance-level image retrieval. On most retrieval benchmarks, deep methods perform worse than conventional methods that rely on local descriptor matching and reranking with elaborate spatial verification (Mikulík et al. 2010; Tolias et al. 2015; Tolias and Jégou 2015; Li et al. 2015).

Most of the deep retrieval methods use networks as local feature extractors, leveraging models pretrained on large image classification datasets such as ImageNet (Deng et al. 2009), and only focus on designing image representations suited for image retrieval on top of these features. Contributions have been made to allow deep architectures to accurately represent input images of different sizes and aspect ratios (Babenko and Lempitsky 2015; Kalantidis et al. 2016; Tolias et al. 2016) or to address the lack of geometric invariance of CNN-based features (Gong et al. 2014; Razavian et al. 2014). Here, we argue that one of the main reasons that prevented previous retrieval methods based on deep architectures to perform well is their lack of supervised learning for the specific task of instance-level image retrieval.

In this work, we focus on the problem of *learning representations that are well suited for the retrieval task*. Unlike features that are learned to distinguish between different semantic categories, and hence that are supposedly robust to intraclass variability, here we are interested in distinguishing between particular objects, even if they belong to the same semantic class. We propose a solution that combines a representation tailored for the retrieval task together with a training procedure that explicitly targets retrieval.

For the representation, we build on the regional maximum activations of convolutions (R-MAC) descriptor (Tolias et al. 2016). This approach computes CNN-based descriptors of several image regions at different scales that are sum-aggregated into a compact feature vector of fixed length, and is therefore moderately robust to scale and translation. An advantage of this method is that it can encode images at high resolutions and without distorting their aspect ratio. However, in its original form, the R-MAC descriptor uses a CNN pretrained on ImageNet, which we believe is sub-optimal. In our work, we note that all the steps of the R-MAC pipeline can be integrated in a single CNN and we propose to learn its weights in an end-to-end manner, as all the steps involved in its computation are differentiable.

For the training procedure, we use a siamese network that combines three streams with a triplet loss and that explicitly optimizes the weights of our network to produce representations well suited for a retrieval task. Furthermore, we also propose to learn the pooling mechanism of the R-MAC descriptor. In the original architecture of Tolias et al. (2016), a rigid grid determines the location of regions that are pooled to produce the final image-level descriptor. Here we propose to explicitly learn how to choose these regions given the image content using a region proposal network. The training procedure results in a novel architecture that is able to encode one image into a compact fixed-length vector in a single forward pass. Representations of different images can be then compared using the dot-product. Finally, we propose a way to encode information at different resolutions into a single descriptor. Input images are first resized at different scales and their representations are then combined, yielding a multi-resolution descriptor that significantly improves the results.

Learning the weights of our representation requires appropriate training data. To that aim we leverage the public Landmarks dataset of Babenko et al. (2014), which is well aligned with the standard instance-level retrieval benchmarks as shown by Babenko et al. (2014), and where images were retrieved by querying image search engines with the name of several famous landmarks. We propose a cleaning process for this dataset that automatically discards the large amount of mislabeled images and estimates the landmark location without the need of further annotations or manual intervention.

An extensive experimental study on four standard image retrieval benchmarks quantitatively evaluates the impact of each of our contributions. We also show the effect of combining our representation with query expansion and database-side feature augmentation, and the impact of compression with product quantization. In the end, we obtain results that largely outperform the state of the art on all datasets, not only compared to methods that use one global representation per image, but also against much more costly methods that, unlike our proposed method, require to perform a subsequent matching stage or geometrical verification.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the cleaning procedure that leads to a suitable training set. Section 4 describes the training procedure while Sect. 5 proposes several improvements to our deep architecture. Section 6 describes the final pipeline and compares it with the state of the art. Finally, Sect. 7 concludes the paper.

This article extends our previous work (Gordo et al. 2016) in the following manner: we consider residual network architectures as an alternative when constructing our global descriptor (and their very deep nature requires to adjust our training procedure, see Sect. 4.3). We build a multi-resolution version of the descriptor to cope with scale changes between query and database images (Sect. 5.3). We propose to combine our method with database-side feature augmentation to significantly improve the retrieval accuracy with no extra cost at testing time (Sect. 6.2). We evaluate the effect of compression in our representation, both with PCA and with product quantization (Sect. 6.3). These new contributions lead to

significantly improved results. Furthermore, we also show qualitative results illustrating the impact of learning in the model activations.

2 Related Work on Image Retrieval

This section gives an overview of some of the key papers that have contributed to instance-level image retrieval.

2.1 Conventional Image Retrieval

Early techniques for instance-level retrieval such as the ones of Sivic and Zisserman (2003), Nister and Stewenius (2006), and Philbin et al. (2007) rely on bag-of-features representations, large vocabularies, and inverted files. Numerous methods that better approximate the matching of the descriptors have been proposed, see for instance the works of Jégou et al. (2008, 2010a), Mikulik et al. (2013), Tolias et al. (2015). An advantage of these techniques is that spatial verification can be employed to rerank a shortlist of results (Philbin et al. 2007; Perdoch et al. 2009), yielding a large improvement despite a significant cost.

Concurrently, methods that aggregate local patches to build a global image representation have been considered. Encoding techniques, such as the Fisher Vector (Perronnin and Dance 2007; Perronnin et al. 2010) or the VLAD descriptor (Jégou et al. 2010b) have been used for example by Perronnin et al. (2010), Gordo et al. (2012), Jégou and Chum (2012), Radenovic et al. (2015). All these methods can be combined with postprocessing techniques such as query expansion (Chum et al. 2007, 2011; Arandjelovic and Zisserman 2012). Some works also suggest to compress the descriptors to improve the storage requirements and retrieval efficiency at the cost of reduced accuracy. Although the most common approach is to use unsupervised compression through PCA or product quantization (Perronnin et al. 2010; Jégou and Chum 2012; Radenovic et al. 2015), supervised dimensionality reduction approaches are also possible (Gordo et al. 2012).

2.2 CNN-Based Retrieval

In the seminal work of Krizhevsky et al. (2012), the activations of a CNN trained for ImageNet classification were used as image features for an instance-level retrieval task, although this was only evaluated in qualitative terms. Soon after, these off-the-shelf CNN features were evaluated quantitatively by Razavian et al. (2014). Several improvements were proposed to overcome their lack of robustness to scaling, cropping and image clutter. The method of Razavian et al. (2014) performs region cross-matching and accumulates the maximum similarity per query region while the one of Babenko and Lempitsky (2015) applies sum-pooling to whitened region descriptors. Kalantidis et al. (2016) extended the work of Babenko and Lempitsky (2015) by allowing cross-dimensional weighting and aggregation of neural codes. Other approaches proposed hybrid models also involving an encoding technique such as Perronnin and Larlus (2015) that used the FV or Gong et al. (2014) and Paulin et al. (2015) that considered VLAD. Although these methods outperform standard global descriptors, their performance is significantly below the state of the art of conventional methods.

Tolias et al. (2016) proposed to aggregate the activation features of a CNN in a fixed layout of spatial regions. The method uses a pretrained, fully convolutional CNN to extract local features of images without distorting their aspect ratio and independently of their size, and aggregates these local features into a global representation using normalizations known to work well for image retrieval (Jégou and Chum 2012). The result is the R-MAC descriptor, a fixed-length vector representation of the image that, when combined with query expansion, achieves results close to the state of the art. Our work draws inspiration from the R-MAC pipeline, but learns the model weights in an end-to-end manner.

2.3 Finetuning for Retrieval

The use of off-the-shelf features from models trained for classification on ImageNet may not be the optimal choice for instance-level retrieval tasks due to the models being trained to achieve intraclass generalization. Instead of using pretrained models as a feature extractor, a few methods have proposed to explicitly learn weights more suited for the retrieval task. The work of Babenko et al. (2014) showed that models pretrained on ImageNet for object classification could be improved by finetuning them on an external set of Landmarks images, even when using a classification loss.

A preliminary version of our work (Gordo et al. 2016), together with a concurrent work (Radenovic et al. 2016), confirmed that finetuning the pretrained models for retrieval can bring a significant improvement, but demonstrated that even more crucial are the combination of i) a good image representation and ii) a ranking loss—as opposed to the classification loss used by Babenko et al. (2014). The recent NetVLAD by Arandjelovic et al. (2016) also highlights the importance of learning to rank.

3 Leveraging Large-Scale Noisy Data

To learn an informative and efficient representation for instance-level retrieval, we need the appropriate dataset.

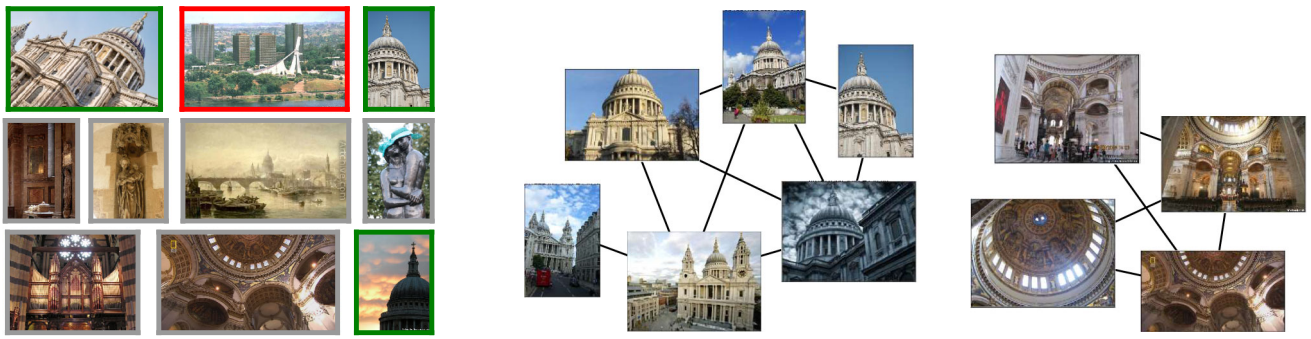

Fig. 1 *Left* random images from the "St Paul's Cathedral" landmark. *Green*, *gray* and *red borders* respectively denote prototypical, non-prototypical, and incorrect images. *Right* excerpt of the two largest connected components of the pairwise matching graph (corresponding to outside and inside pictures of the cathedral) (Color figure online)

This section describes how we leveraged and automatically cleaned an existing dataset to obtain the characteristics we need for training our models.

We leverage the **Landmarks** dataset (Babenko et al. 2014), a large-scale image dataset that contains approximately 214k images of 672 famous landmark sites. Its images were collected through textual queries in an image search engine without thorough verification. As a consequence, they comprise a large variety of profiles: general views of the site, close-ups of details like statues or paintings, with all intermediate cases as well, but also site map pictures, artistic drawings, or even completely unrelated images, see Fig. 1.

We could only download a subset of all images due to broken URLs. We removed classes with too few images. We also meticulously removed all classes having an overlap with the Oxford 5k, Paris 6k, and Holidays datasets, on which we experiment, see Sect. 4.4. We obtained a set of about 192,000 images divided into 586 landmarks. We refer to this set as **Landmarks-full**. For our experiments, we use 168,882 images for the actual finetuning, and the 20,668 remaining ones to validate parameters.

Cleaning the Landmarks Dataset. The Landmarks dataset presents a non-negligible amount of unrelated images (Fig. 1). While this could be allowed in certain frameworks (*e.g.* for classification, typically networks can accommodate during training for this diversity and even for noise), in some scenarios we need to learn our representations with images of the *same* particular object or scene. In this case, variability comes from different viewing scales, angles, lighting conditions and image clutter. We preprocess the Landmarks dataset to achieve this as follows.

We first run a strong image matching baseline within the images of each landmark class. We compare each pair of images using invariant keypoint matching and spatial verification (Lowe 2004). We use the SIFT and Hessian-Affine keypoint detectors (Lowe 2004; Mikolajczyk and Schmid 2004) and match keypoints using the first-to-second neighbor ratio rule (Lowe 2004). This is known to outperform approaches based on descriptor quantization (Philbin et al. 2010). Afterwards, we verify all pairwise matches with an affine transformation model as proposed by Philbin et al. (2007). This heavy procedure is affordable as it is performed offline, only once at training time, and on a class per class basis.

Without loss of generality, we describe the rest of the cleaning procedure for a single landmark class. Once we have obtained a set of pairwise scores between all image pairs, we construct a graph whose nodes are the images and edges are pairwise matches. We prune all edges which have a low score. Then we extract the connected components of the graph. They correspond to different profiles of a landmark; see Fig. 1 that shows the two largest connected components for St Paul's Cathedral. Finally we retain only the largest connected component and discard the others to ensure that all images inside a class are visually related. This cleaning process leaves about 49,000 images (divided in 42,410 training and 6382 validation images) still belonging to one of the 586 landmarks, referred to as **Landmarks-clean**. The cleaning process took approximately 1 week on a 32-core server, parallelizing over classes.

Bounding Box Estimation. In one of our experiments, we replace the uniform sampling of regions in the R-MAC descriptor by a learned region of interest (ROI) selector (Sect. 5.1). This selector is trained using bounding box annotations that we automatically estimate for all landmark images. To that aim we leverage the data obtained during the cleaning step. The position of verified keypoint matches is a meaningful cue since the object of interest is consistently visible across the landmark's pictures, whereas distractor backgrounds or foreground objects are varying and hence unmatched.

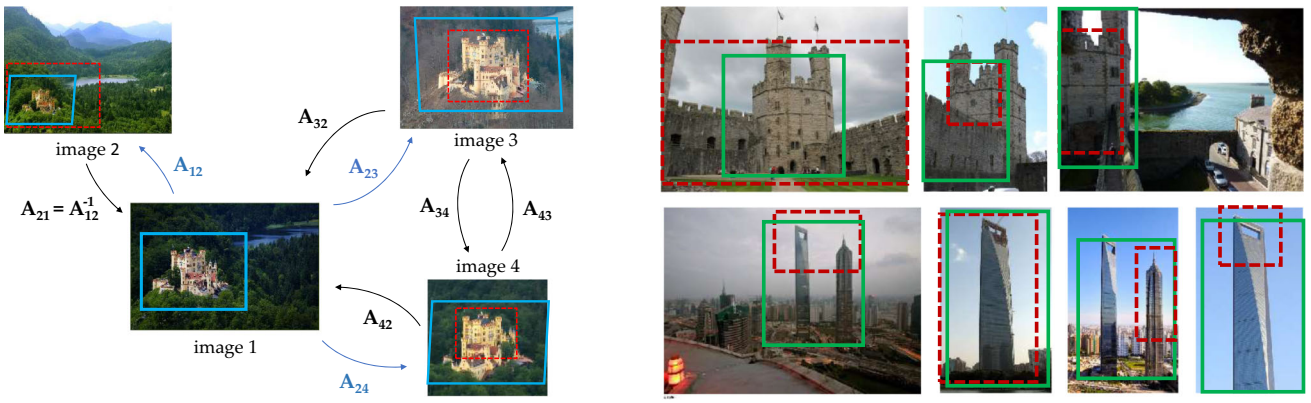

Fig. 2 *Left* the bounding box from image 1 is projected into its graph neighbors using the affine transformations (*blue rectangles*). The current bounding box estimates (*dotted red rectangles*) are then updated accordingly. The diffusion process repeats through all edges until convergence. *Right* initial (*dotted red box*) and final (*solid green box*) estimates (Color figure online)

We denote the connected component from each landmark as a graph $\mathcal{S} = \{\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S}\}$. Each pair of connected images $(i, j) \in \mathcal{E}_\mathcal{S}$ corresponds to a set of verified keypoint matches and an affine transformation A_{ij}. We first define an initial bounding box in both images i and j, denoted by B_i and B_j, as the minimum rectangle enclosing all matched keypoints. Note that a single image can be involved in many different pairs. In this case, the initial bounding box is the geometric median of all boxes, efficiently computed as in Vardi and Zhang (2004). Then, we run a diffusion process, illustrated in Fig. 2, in which for a pair (i, j) we predict the bounding box B_j using B_i and the affine transform A_{ij} (and conversely). At each iteration, bounding boxes are updated as: $B'_j = (\alpha - 1)B_j + \alpha A_{ij} B_i$, where α is a small update step (we set $\alpha = 0.1$ in our experiments). Again, the multiple updates for a single image are merged using geometric median, which is robust against poorly estimated affine transformations. This process iterates until convergence. As can be seen in Fig. 2, the locations of the bounding boxes are improved as well as their consistency across images. We are making the list of Landmarks-clean images and the estimated bounding boxes available.

Next we leverage our cleaned dataset to learn powerful image representations tailored for image retrieval.

4 Learning to Rank: An End-to-End Approach

This section first revisits the R-MAC representation of Tolias et al. (2016) in Sect. 4.1 and shows that, despite its hand-crafted nature, all the operations involved in it can be integrated into a single CNN that computes the R-MAC representation in one single forward pass. More importantly, all of its components consist of differentiable operations, and therefore, given training data and an appropriate loss, one can learn the optimal weights of the architecture in an end-to-end manner. To that aim we leverage a three-stream siamese network with a triplet ranking loss (Sect. 4.2). Then we discuss the practical details that allow this architecture to scale to deep networks with large memory needs (Sect. 4.3). Finally, we experimentally validate the gain obtained by the proposed training strategy in terms of accuracy in standard benchmarks (Sect. 4.4).

4.1 The R-MAC Baseline

The R-MAC descriptor, recently introduced by Tolias et al. (2016), is a global image representation that is particularly well-suited for image retrieval. At its core, it uses a "fully convolutional" CNN as a powerful local feature extractor that works independently of the image size and that extracts local features without distorting the aspect ratio of the original image. The original work of Tolias et al. (2016) uses both AlexNet (Krizhevsky et al. 2012) and VGG16 (Simonyan and Zisserman 2015) network architectures, with models pretrained on the ImageNet dataset, but other network architectures such as residual networks (He et al. 2016) can also be used. These local features are then max-pooled across several multi-scale overlapping regions, obtained from a rigid grid covering the image, similar in spirit to spatial pyramids, producing a single feature vector per region. These region-level features are independently ℓ_2-normalized, whitened with PCA, and ℓ_2-normalized again, a normalization pipeline known to work well for image retrieval (Jégou and Chum 2012). Finally, region descriptors are sum-aggregated and ℓ_2-normalized once again. The obtained global image representation is a compact vector whose size (typically 256 to 2k dimensions, depending on the network architecture) is independent of the size of the image and of the number of regions. Note that the region pooling is different from a spatial pyramid: the latter concatenates the region descriptors, while the for-

mer sum-aggregates them. Comparing the R-MAC vectors of two images with a dot-product can then be interpreted as a weighted many-to-many region matching, where the weights depend on the norm of the aggregated region descriptors.

4.2 Learning to Retrieve

One key aspect of the R-MAC pipeline is that all of its components are differentiable operations. More precisely, the multi-scale spatial pooling in different regions is equivalent to the Region of Interest (ROI) pooling from He et al. (2014) using a fixed rigid grid, which is differentiable as shown in the context of detection (Girshick 2015). The PCA projection can be seen as a combination of a shifting (for the mean centering) and a fully connected (FC) layer (for the projection with the eigenvectors), with weights that can be learned. The sum-aggregation of the different regions and the ℓ_2-normalization are also differentiable. Therefore, one can implement a network architecture that, given an image and the precomputed coordinates of its regions, directly produces a representation equivalent to the R-MAC pipeline. As all the components are differentiable, one can backpropagate through the network architecture to learn the optimal network weights, namely the weights of the convolutions and of the shifting and fully-connected layers that replace the PCA.

Learning with a Classification Loss. One can easily finetune a standard classification architecture (*e.g.* VGG16) on the Landmarks dataset using a cross-entropy loss, as previously done by Babenko et al. (2014), and then use the improved convolutional filters as the feature extractor of the R-MAC pipeline, instead of using the original weights. We use this approach as our training baseline, and note that it has important issues. First, it does not learn directly the task to address, retrieval, but a proxy, classification. Second, it does not leverage the R-MAC architecture, as it learns on the original classification architecture, using low-resolution square crops. The convolutional weights are used together with the R-MAC architecture only after the training has finished. In our experiments we show how this naive finetuning method already outperforms the baseline approach significantly, but does not match the accuracy obtained by training using the appropriate architecture and loss.

Learning with a Ranking Loss. In our work we propose to consider a ranking loss based on image triplets. The goal is to explicitly enforce that, given a triplet composed of a query image, a relevant element to the query, and an irrelevant one, the R-MAC representation of the relevant image is closer to the representation of the query than the representation of the irrelevant one.

We design a three-stream siamese network architecture where the image representation produced by each of the three streams are jointly considered by the loss. This architecture is illustrated in Fig. 3. The weights of the convolutional filters and of the fully-connected layer are shared between the streams as their size is independent of the size of the images. This means that the siamese architecture can process images of any sizes and aspect ratios, and we can train the network using images at the same (high) resolution that is used at test time.

Siamese networks have performed well for metric learning (Song et al. 2016), dimensionality reduction (Hadsell et al. 2006), learning image descriptors (Simo-Serra et al. 2015), and performing face identification (Chopra et al. 2005; Hu et al. 2014; Sun et al. 2014). Recently triplet networks (i.e. three-stream siamese networks) have been considered for metric learning (Hoffer and Ailon 2015; Wang et al. 2014) and face identification (Schroff et al. 2015).

We use the following ranking loss. Let I_q be a query image with R-MAC descriptor q, I^+ be a relevant image with descriptor d^+, and I^- be an irrelevant image with descriptor d^-. We define the ranking triplet loss as

$$L(I_q, I^+, I^-) = \frac{1}{2}\max(0, m + \|q - d^+\|^2 - \|q - d^-\|^2), \tag{1}$$

where m is a scalar that controls the margin. Given a triplet that produces a non-zero loss, the sub-gradients are given by:

$$\frac{\partial L}{\partial q} = d^- - d^+, \quad \frac{\partial L}{\partial d^+} = d^+ - q, \quad \frac{\partial L}{\partial d^-} = q - d^-. \tag{2}$$

The sub-gradients are backpropagated through the three streams of the network, and the convolutional layers together with the "PCA" layers—the shifting and the fully connected layers—get updated. This approach directly optimizes a ranking objective.

4.3 Practical Considerations

When learning with a ranking loss, one should pay attention to certain practical considerations. The first one is the sampling of the triplets, as sampling them randomly will, most of the time, yield triplets that incur no loss and therefore do not improve the model. To ensure that the sampled triplets are useful, we first select randomly N training samples, extract their features with the current model, and compute all possible triplets and their losses, which is fast once the features have been extracted. All the triplets that incur a loss are preselected as good candidates. Triplets can then be sampled

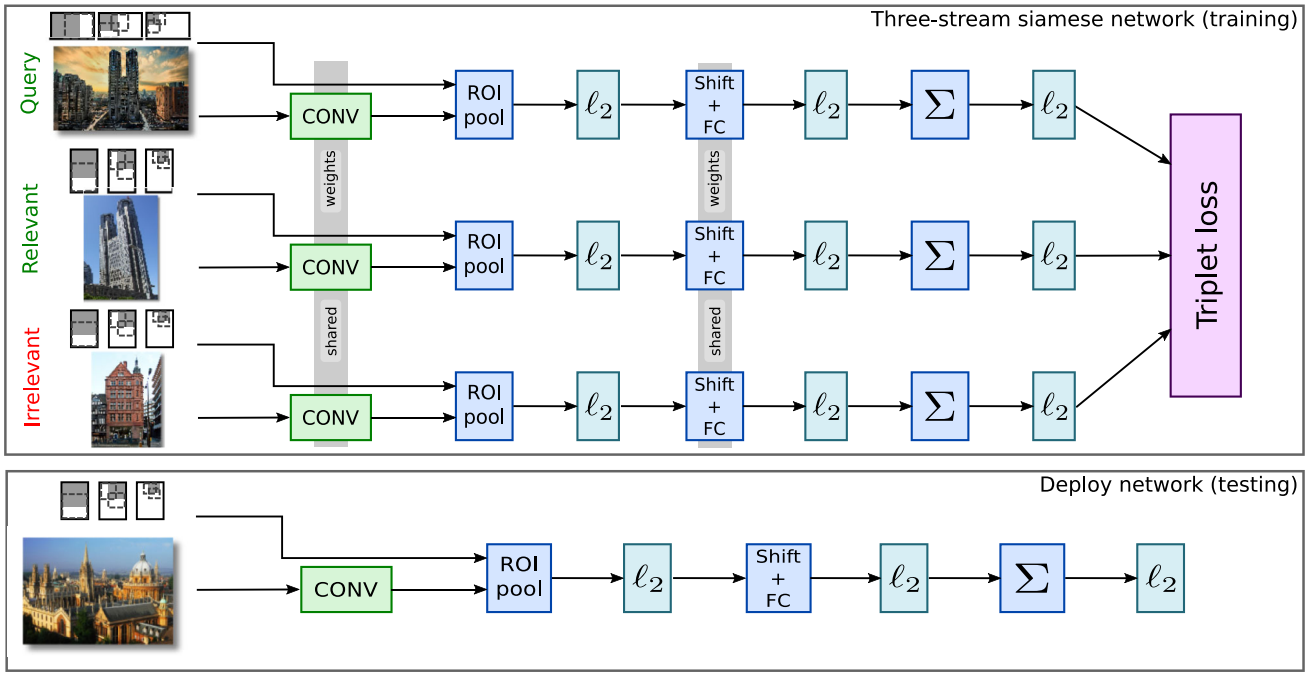

Fig. 3 *Proposed siamese network* At training time, image triplets are sampled and simultaneously considered by a *triplet-loss* that is well-suited for the task (*top*). At test time, the query image is fed to the learned architecture to efficiently produce a *compact global image representation* that can be compared with the dataset image representations with a dot-product (*bottom*)

from that set of good candidates, with a bias towards hard triplets, *i.e.* triplets that produce a high loss. In practice this is achieved by randomly sampling one of the N images with a uniform distribution and then randomly choosing one of the 25 triplets with the largest loss that involve that particular image as a query. Note that, in theory, one should recompute the set of good candidates every time the model gets updated, which is very time consuming. In practice, we assume that most of the hard triplets for a given model will remain hard even if the model gets updated a few times, and therefore we only update the set of good candidates after the model has been updated k times. We used $N = 5000$ samples and $k = 64$ iterations with a batch size of 64 triplets per iteration in our experiments.

The second consideration is the amount of memory required during training, as we train with large images (larger side resized to 800 pixels) and with three streams at the same time. When using the VGG16 architecture, we could only fit one triplet in memory at a time on an M40 GPU with 12 GB of memory. To perform updates with a batch of effective size bs larger than one, we sequentially compute and aggregate the gradients of the loss with respect to the parameters of the network for every triplet, and only perform the actual update every bs triplets, with $bs = 64$.

When using a larger network such as ResNet101, the situation becomes more complex, as we do not have enough memory to process even one single triplet. Instead of reducing the image size, which would result in a loss of detail, we propose an alternative approach detailed in Algorithm 1. This approach allows us to process the streams of a triplet sequentially using one single stream instead of all of them

Algorithm 1 Memory efficient model update

procedure PROCESS TRIPLET
  Q: Query image
  I^+: Relevant image
  I^-: Irrelevant image
Main:
  Compute feature representation of Q: q
  Compute feature representation of I^+: d^+
    /Overwrites results needed to backpropagate the loss with respect to q/
  Compute feature representation of I^-: d^-
    /Overwrites results needed to backpropagate the loss with respect to d^+/
  Compute loss as in Eq. (1)
  Compute gradients with respect to q, d^+, and d^- as in Eq. (2)
  Backpropagate the loss with respect to d^-
  Recompute q
    /recomputing is needed to obtain the necessary statistics to backpropagate/
  Backpropagate the loss with respect to q
  Recompute d^+
    /recomputing is needed to obtain the necessary statistics to backpropagate/
  Backpropagate the loss with respect to d^+

simultaneously. This yields exactly the same gradients but trades some computational efficiency due to recomputations (about a 25% overhead) for very significant memory reduction (only one third of the memory is required, from 23 Gb down to 7.5 Gb). This allows one to train the model using very deep architectures without reducing the size of the training images.

4.4 Experiments

In this section we study the impact of learning the weights for different setups and architectures. In all these experiments we assume that the descriptor extracts regions following the standard R-MAC strategy, i.e. following a predefined rigid grid.

4.4.1 Experimental Details

We test our approach on four standard datasets: the *Oxford* 5k building dataset (Philbin et al. 2007), the *Paris* 6k dataset (Philbin et al. 2008), the INRIA *Holidays* dataset (Jégou et al. 2008), and the University of Kentucky Benchmark *(UKB)* dataset (Nister and Stewenius 2006). We use the standard evaluation protocols, *i.e.* recall@4 for UKB and mean average precision (mAP) for the rest. As is standard practice, in Oxford and Paris one uses only the annotated region of interest of the query, while for Holidays and UKB one uses the whole query image. Furthermore, the query image is removed from the dataset when evaluating on Holidays, but not on Oxford, Paris, and UKB. Following most CNN-based methods, we manually correct the orientation of the images on the Holidays dataset and evaluate on the corrected images. For fair comparison with methods that do not correct the orientation we also report results without correcting the orientation in our final experiments.

For the convolutional part of our network, we evaluate two popular architectures: VGG16 (Simonyan and Zisserman 2015) and ResNet101 (He et al. 2016). In both cases we start with the publicly available models pretrained on the ImageNet ILSVRC data. The fully-connected layer is initialized with a PCA projection, computed on the normalized per-region descriptors. All subsequent learning is performed on the Landmarks dataset.

To perform finetuning with classification we follow standard practice and resize the training images to multiple scales (shortest side in the [256–512] range) and extract random crops of 224×224 pixels. To finetune using our proposed architecture we also augment our training data performing random crops (randomly removing up to 5% of each side of the image) and then resize the resulting crop such as that the larger side is of 800 pixels, preserving the aspect ratio. At test time, all the database images are also resized so the larger side is 800 pixels.[1] All the models are trained with stochastic gradient descent (SGD) with momentum of 0.9, learning rate of 10^{-3}, and weight decay of 5×10^{-5}. We decrease the learning rate down to 10^{-4} on the classification finetuning once the validation error on Landmarks stops decreasing. We did not see any improvement by reducing the learning rate when learning to rank, and so we keep the learning rate at 10^{-3} until the end. The margin is set to $m = 0.1$.

4.4.2 Results

Quantitative Evaluation. We report results in Table 1 for two possible choices of the convolutional part of the network: VGG16 (top) and ResNet101 (bottom). For each architecture, we first report performance with the R-MAC baseline, whose convolutional layer weights are taken directly from the ImageNet pretrained networks and the PCA is learned on Landmarks-full. For the learned models, weights are finetuned on Landmarks either with a classification loss (**Ft-Cls**) or with a ranking loss (**Ft-Rnk**). For the latter, we consider either initializing the weights directly with the ImageNet pretrained network or with a warmed up model already finetuned on Landmarks using a classification loss.

From the results reported in Table 1 we highlight the following observations.

- Finetuning with a naive classification loss on a relevant dataset already brings a significant improvement over a model pretrained on ImageNet, as already observed by Babenko et al. (2014) (albeit on a different architecture) on the first three datasets. In this case, training with Landmarks-full or training with Landmarks-clean does not make a significant difference.
- Finetuning our proposed architecture with a ranking loss is the best performing strategy. For the first three datasets again, it seems very beneficial to improve the weights of our model using the Landmarks dataset. We only report results learning the ranking with Landmarks-clean. We found this to be crucial: learning on Landmarks-full significantly worsens the accuracy of the model.
- To obtain good results with VGG16 using the ranking loss we found important to warm up the network by first training it on the Landmarks dataset using a classification loss, as done in Gordo et al. (2016). However, this was not so important when using the more recent ResNet101 architecture: although warming up the network brings slight improvements, the final results are similar. This can also be observed in Fig. 4, which shows the evolution of

[1] Note that this differs from the original setup of Tolias et al. (2016), that resizes images to 1024 pixels, and leads to different results in Table 1. Please see Gordo et al. (2016) for a discussion about this issue.

Table 1 Impact of learning the weights of the representation with a classification (Cls) and a ranking (Rnk) loss, either with VGG16 or ResNet101

Architecture	Model	Oxford 5k	Paris 6k	Holidays	UKB
VGG16	ILSVRC2012 baseline	60.3	79.9	85.8	3.75
	Ft Cls-Landmarks-Full	74.2	82.5	87.7	3.65
	Ft Cls-Landmarks-Clean	74.0	83.0	86.0	3.62
	Ft Rnk-Landmarks-Clean	76.3	86.2	85.6	3.61
	Ft Cls-Landmarks-Full ⇒ Ft Rnk-Landmarks-Clean	79.9	85.9	87.9	3.59
	Ft Cls-Landmarks-Clean ⇒ Ft Rnk-Landmarks-Clean	79.0	86.9	86.4	3.55
ResNet101	ILSVRC2012 baseline	69.4	85.2	91.4	**3.89**
	Ft Cls-Landmarks-Full	77.7	89.4	93.4	**3.89**
	Ft Cls-Landmarks-Clean	78.5	88.2	93.0	3.86
	Ft Rnk-Landmarks-Clean	83.4	92.8	93.7	3.85
	Ft Cls-Landmarks-Full ⇒ Ft Rnk-Landmarks-Clean	**84.1**	**93.6**	**94.0**	3.83
	Ft Cls-Landmarks-Clean ⇒ Ft Rnk-Landmarks-Clean	83.3	91.3	93.3	3.79

Bold values indicate the best results per dataset
The weights are learned either from the full Landmarks dataset (Landmarks-Full) of the clean version (Landmarks-Clean). For the ranking loss we also compare different intializations

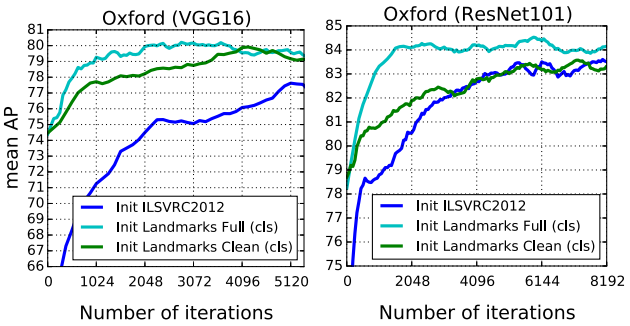

Fig. 4 Accuracy comparison of three different model initializations when finetuning the representation with a ranking loss as a function on the number of iterations (one iteration corresponds to a batch of 64 triplets)

the accuracy on the Oxford dataset as training progresses for different model initializations.

- For the UKB dataset, the "off-the-shelf" R-MAC already provides state-of-the-art results, and the training slightly decreases its performance, probably because of the large domain differences (*c.f*. Sect. 6 for a more detailed discussion about UKB).
- As expected, the model based on ResNet101 outperforms the model based on VGG16. This gap, however, is not as significant as the improvement brought by the training.

Impact of Finetuning on the Neurons. We qualitatively evaluate the impact of finetuning the representation for retrieval. To this end, we visualize the image patches that most strongly respond (i.e. with the largest activation values) for different neurons of the last convolutional VGG16 layer in Fig. 5, before and after finetuning for retrieval. These examples illustrate the process that takes place during finetuning. Some neurons that were originally specialized in recognizing specific object parts crucial for classification on ImageNet (for instance a "shoulder neuron" or a "waist neuron") were repurposed to fire on visually similar landmark parts (*e.g*. domes, buildings with flat roofs and two windows). However, other neurons (*e.g*. the "sunglasses neuron") were not clearly repurposed, which suggest that improvements in the training scheme may be possible.

Computational Cost. To train and test our models we use an M40 NVIDIA GPU with 12 Gb of memory. When pretrain-

Fig. 5 Visualization of the neuron adaptation during training. Image patches with largest activation values for some neurons of layer "conv5_3" from VGG16, before (*top*) and after (*bottom*) finetuning for retrieval. See text for more details

ing ResNet101 on Landmarks-full with a classification loss it takes approximately 4 days to perform 80,000 iterations with a batch size of 128. This is the model that we use to initialize our approach in most of the experiments. For training our ranking model we use a batch size of 64 triplets and the "single stream" approach (Algorithm 1), and resize our images preserving their aspect ratio such that the longer size has 800 pixels. With ResNet101 this process requires about 7.5 GB of memory per stream per sample, and can process 64 iterations in approximately 1 h, of which approximately 15 min are devoted to mining hard triplets. Our model, when initialized with the pretrained classification model, converges after approximately 3000 iterations, *i.e.*, 2 days. If we do not warm up the model on Landmarks and use directly the ImageNet model, it converges after approximately 8000 iterations. In both cases, this roughly corresponds to a week of total training.

Once trained, extracting the descriptor of one image takes approximately 150 ms, *i.e.*, about 7 images per second on a single GPU. Computing the similarity between two images comes down to computing the dot-product between their representations, which is very efficient, *i.e.*, one can compute millions of such comparisons per second on a standard processor.

5 Improving the R-MAC Representation

The R-MAC representation has proved to excel at retrieval among deep methods (Tolias et al. 2016). In the previous section we have shown that we could further improve its effectiveness by learning the network weights in an end-to-end manner with an objective and a training set tailored for image retrieval. In this section, we propose several ways to modify the network architecture itself. First, we improve the region pooling mechanism by introducing a region proposal network (RPN) that predicts the most relevant regions of the image, where the local features should be extracted (Sect. 5.1). Second, we observe that the network architecture only considers a single fixed image resolution, and propose to extend it to build a multi-resolution descriptor (Sect. 5.2).

5.1 Beyond Fixed Regions: Proposal Pooling

The rigid multi-scale grid used in R-MAC to pool regions tries to ensure that the object of interest is covered by at least one of the regions. However, this raises two issues. First, it is unlikely that any of the grid regions precisely align with the object of interest. Second, many of the regions only cover background, especially if the object to retrieve is of small scale. This is problematic as the comparison between R-MAC signatures can be seen as a many-to-many region matching, and so region clutter will negatively affect the performance. Increasing the number of regions in the grid would improve the likelihood that one region is well aligned with the object of interest, but would also increase the number of irrelevant regions.

We propose to modify the R-MAC architecture to enhance it with the ability to focus on relevant regions in the image. To this end we replace the rigid grid with a region proposal network (RPN) trained to localize regions of interest in images, similar to the proposal mechanism of Ren et al. (2015). This RPN is trained using the approximate bounding box annotations of the Landmarks dataset obtained as a by-product of our cleaning process. The resulting network architecture is illustrated in Fig. 6.

The main idea behind an RPN is to predict, for a set of candidate boxes of various sizes and aspect ratios, a score describing how likely each candidate box at each possible image location contains an object of interest. Simultaneously, for each candidate box, it performs coordinate regression to improve the location accuracy. This is achieved by a "fully convolutional" network consisting of a first layer that uses 3×3 filters, and two sibling convolutional layers with 1×1 filters that predict, for each candidate box in the image and for each location, both the objectness score and the regressed coordinates. Non-maximum suppression is then performed on the ranked boxes to produce k final proposals per image that are used to replace the rigid grid.

This modification to the network has several positive outcomes. First, the region proposals typically cover the object of interest more tightly than the rigid grid. Second, even if they do not overlap exactly with the region of interest, most of the proposals do overlap significantly with it, which means

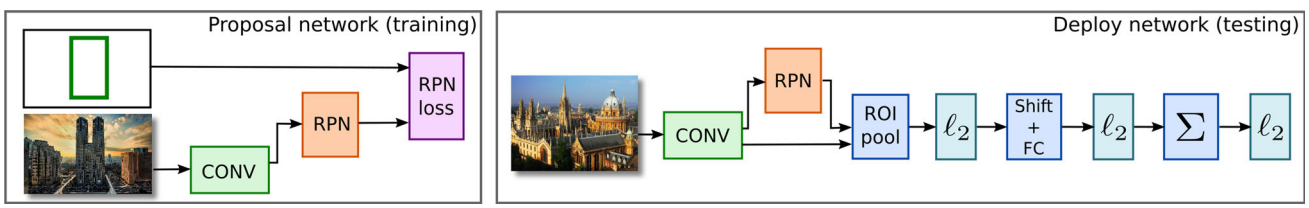

Fig. 6 *Proposal network* At train time, a region proposal network is trained using bounding box annotations and an appropriate loss (*left*). At test time, the query image is fed to the learned architecture to efficiently produce a *compact global image representation* that can be compared with the dataset image representations with a simple dot-product (*right*)

that increasing the number of proposals per image not only helps to increase the coverage but also helps in the many-to-many matching.

Learning the RPN. We assign a binary class label to each candidate box, depending on how much this box overlaps with the ground truth region of interest, and we minimize an objective function with a multitask loss that combines a classification loss (more precisely a log loss over object *vs* background classes) and a regression loss [similar to the smooth ℓ_1 loss used by Girshick (2015)]. The objective function is optimized by backpropagation and SGD. More details about the implementation and the training procedure of the RPNs can be found in the work of Ren et al. (2015).

We learn the RPN on top of the convolutional layers of our network. We first train the network using the rigid grid as described in Sect. 4.2, and then we fix the weights of the convolutional layers and train the RPN from the output of the last convolutional layer. In this way, both networks share the computation of the convolutional part and are combined into a single architecture (Fig. 6). Finetuning the RPN *and* the ranking simultaneously is also feasible, but we observed no accuracy gain by doing so.

5.2 Multi-Resolution

In the original R-MAC descriptor as proposed by Tolias et al. (2016), images are considered at a single scale. However, one could consider extracting and combining features from images that have been resized to different resolutions, in order to integrate information from different scales. The goal is to improve the matching between objects that appear at different scales in the database images and the retrieval of small objects.

One interesting characteristic of the original R-MAC network is that different input image sizes still produce descriptors of the same length. Note, however, that two versions of the same image with different resolutions will *not* produce the same output descriptor. The first part of the network is fully convolutional, which directly enables to process inputs of different sizes, and the aggregation layer combines the size-dependent amount of input features into a fixed-length representation. Following this idea, we propose to extract different descriptors from images resized at different scales, and then combine them into a single final representation. In practice we use 3 scales, with 550, 800 and 1050 pixels in the larger side, preserving the aspect ratio. The descriptor of each of the three images is then extracted independently. Finally we sum-aggregate them and ℓ_2-normalize them to obtain the final descriptor.

This multi-resolution descriptor can be computed both in the query side and in the database side. The process brings an extra computational cost at feature extraction time (approximately three times the cost for three resolutions), but the cost at search time and the storage cost remain the same.

Our multi-resolution scheme can be connected to previous papers aiming to build transformation-invariant representations like Schmidhuber (2012), Laptev et al. (2016). The transformation considered in our case is the image scaling. In contrast to multi-column networks or bagging approaches (Schmidhuber 2012), we use the same network for all image scales. In fact, our approach is conceptually close to Laptev et al. (2016), a siamese network with weight sharing, the main difference being that we use average-pooling instead of max-pooling.

5.3 Experiments

In this section we study the impact of the proposal pooling and the multi-resolution descriptors.

Experimental Details. We train the RPN network for 200k iterations with a weight decay of $5 \cdot 10^{-5}$ and a learning rate of 10^{-3}, which is decreased by a factor of 10 after 100k iterations. We remark that only the RPN layers are updated and that the preceding convolutional layers remain fixed. The process takes less than 12 h on an M40 GPU.

Region Proposal Network. Table 2 presents the results of the region proposal network for an increasing number of regions compared to a rigid grid both for the baseline R-MAC (convolution weights learned from ImageNet) and for the version trained with a ranking loss, for both VGG16 and ResNet101 architectures. With VGG16 we observe a significant improvement for all datasets and types of training when the number of regions is high enough (128 regions or more), consistent with our findings in the preliminary version of this article (Gordo et al. 2016). However, with ResNet101, this gap is much smaller, especially when the network has been trained with the ranking loss. Our intuition is that ResNet101 is able to learn a more invariant representation of the regions and to discount the effect of background, and so it does not require the proposals as much as VGG16. This makes the use of proposals less appealing when using ResNet101. Given that ResNet101 considerably outperforms VGG16 for all the cases (*c.f.* Tables 1, 2), we depart from Gordo et al. (2016) and, for the rest of the paper, we report results only with ResNet101 without using the RPN.

Multi-Resolution. Table 3 shows results using ResNet101 trained with a ranking loss. Multi-resolution is applied to the query image (QMR), to the database images (DMR), or to

Table 2 Accuracy comparison between the fixed-grid and our proposal network, for an increasingly large number of proposals, before and after finetuning with a ranking-loss

Dataset	Model	Grid	# Region proposals					
			20	32	64	128	192	256
VGG16								
Oxford 5k	ILSVRC2012 baseline	60.3	62.4	63.1	63.3	64.3	65.0	**65.4**
	Ft Cls-Full ⇒ Ft Rnk-Clean	79.9	80.7	80.8	81.9	83.1	**83.2**	**83.2**
Paris 6k	ILSVRC2012 baseline	79.9	77.6	78.5	79.7	80.6	81.1	**81.3**
	Ft Cls-Full ⇒ Ft Rnk-Clean	85.9	85.1	85.7	86.6	87.1	87.1	**87.2**
Holidays	ILSVRC2012 baseline	85.8	82.7	83.5	85.8	86.8	**87.5**	**87.5**
	Ft Cls-Full ⇒ Ft Rnk-Clean	87.9	86.2	86.7	87.8	**88.7**	**88.7**	**88.7**
UKB	ILSVRC2012 baseline	3.75	3.72	3.74	3.76	3.77	**3.78**	**3.78**
	Ft Cls-Full ⇒ Ft Rnk-Clean	3.59	3.55	3.58	3.60	3.61	**3.62**	**3.62**
ResNet101								
Oxford 5k	ILSVRC2012 baseline	69.4	69.2	70.5	71.4	72.3	72.5	**72.9**
	Ft Cls-Full ⇒ Ft Rnk-Clean	84.1	83.7	84.1	84.4	85.0	**85.2**	**85.2**
Paris 6k	ILSVRC2012 baseline	85.2	84.5	85.2	86.0	86.3	86.5	**86.6**
	Ft Cls-Full ⇒ Ft Rnk-Clean	93.6	93.3	93.7	**94.0**	**94.0**	**94.0**	**94.0**
Holidays	ILSVRC2012 baseline	91.4	89.8	91.0	92.0	92.2	**92.3**	92.2
	Ft Cls-Full ⇒ Ft Rnk-Clean	94.0	92.0	92.7	93.5	93.8	**94.0**	**94.0**
UKB	ILSVRC2012 baseline	3.89	3.89	3.89	3.90	3.90	**3.90**	**3.90**
	Ft Cls-Full ⇒ Ft Rnk-Clean	3.83	3.82	3.82	3.83	3.83	**3.84**	3.83

Bold values indicate the best results for each dataset and model
The rigid grid extracts, on average, 20 regions per image

Table 3 Multi-resolution

QMR	DMR	Oxford 5k	Paris 6k	Holidays	UKB
		84.1	93.6	94.0	3.83
✓		84.9	94.1	94.3	3.83
	✓	85.2	94.1	94.4	3.83
✓	✓	**86.1**	**94.5**	**94.8**	**3.84**

Bold values indicate the best results per dataset
Effect of using multi-resolution descriptors on the query side (QMR) and on the database side (DMR)

both of them. All cases improve over the single-resolution descriptors, showing that encoding images using several scales helps at matching and retrieving objects. QMR and DMR also appear to be complementary. We use both QMR and DMR through the rest of our experiments.

6 Evaluation of the Complete Approach

In the previous sections we have cast the R-MAC descriptor as a standalone network architecture where its weights can be learned discriminatively in an end-to-end manner as well as proposed some improvements over the original pipeline. In this section we compare the obtained representation with the state of the art. Our final method integrates two other improvements: query expansion (QE) and database-side feature augmentation (DBA).

6.1 Query Expansion

To improve the retrieval results we use query expansion, a standard technique introduced to the image search problem by Chum et al. (2007). Query expansion works as follows: a first query is issued with the representation of the query image, and the top k results are retrieved. Those top k results may then undergo a spatial verification stage, where results that do not match the query are discarded. The remaining results, together with the original query, are then sum-aggregated and renormalized. Finally, a second query is issued with the combined descriptor, producing the final list of retrieved images. Query expansion typically leads to large improvements in accuracy at the expense of two extra costs at query time: spatial verification, and a second querying operation. In our case we do not perform spatial verification (note that this typically requires access to local keypoint descriptors, which we do not have), and therefore query expansion simply doubles the query time due to the second query operation.

6.2 Database-Side Feature Augmentation

Introduced in the works of Turcot and Lowe (2009) and Arandjelovic and Zisserman (2012), database-side augmentation (DBA) replaces every image signature in the database

by a combination of itself and its neighbors, potentially after a spatial verification stage as in the case of query expansion. The objective is to improve the quality of the image representations by leveraging the features of their neighbors. Since we do not use spatial verification, we sum-aggregate the nearest k neighbors as in the query expansion case. Optionally, the sum can be weighted depending on the rank of the neighbors, and in our experiments we use weight$(r) = \frac{k-r}{k}$ as a weighting scheme, with r the rank of the neighbor, and k the total number of considered neighbors.

DBA is less common than query expansion as, with sparse inverted files, it increases the size of the database as well as the query time. In our case, signatures are already dense, so we are not affected by this. Consequently, the only extra cost incurs in finding the nearest neighbors in the dataset, which is done only once, and offline. In the case of growing databases, the database augmentation could potentially also be done online as new samples are added.

6.3 Experiments

6.3.1 Evaluation of QE and DBA

We evaluate the effect of query expansion (QE) depending on the number of neighbors k as well as the effect of database-side augmentation (DBA) depending on the number of neighbors k' in Fig. 7. First of all we observe how, in Oxford 5k, where many queries have very few relevant items (less than 10 or even less than 5), using large values of k for the QE can, unsurprisingly, degrade the accuracy instead of improving it, independently of whether DBA is used or not. This is not a problem on Paris, where all queries have a large number of relevant items in the dataset.

The weighted DBA seems to help in all cases, even when large values of k' are selected, but, as a side effect, it can worsen the results as well if an inappropriate number of neighbors are chosen for QE. In general it seems that QE and DBA can significantly help each other if the appropriate number of neighbors is chosen, and, as a rule of thumb, we suggest to use a large value for DBA (*e.g.* $k' = 20$) and a small value for QE (*e.g.* $k = 1$ or $k = 2$). Because DBA can be a costly preprocessing, it is not always feasible. In this case (corresponding to $k' = 0$), it is preferable to use an intermediate value for k. For our final experiments involving QE and DBA, we fix $k = 1$ and $k' = 20$ in all datasets. When employing only QE we fix $k = 10$ in all datasets. If one has prior knowledge about the dataset, modifying these values may lead to improved results.

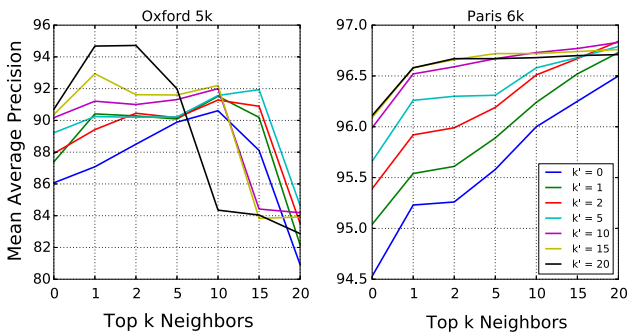

Fig. 7 Accuracy as a function of the number of neighbors k used during query expansion (QE) for several values of the number of neighbors k' used for database-side augmentation (DBA)

6.3.2 Comparison with the State of the Art

We compare our method against the state of the art in Table 4. For these experiments, in addition to the four datasets introduced in Sect. 4.4, we also consider the **Oxford 105k** and **Paris 106k** datasets that extend Oxford 5k and Paris 6k with 100k distractor images (Philbin et al. 2007). In the first half of the table, we show results for other methods that employ global representations of images and do not perform any form of spatial verification or query expansion at run-time. As such, they are conceptually closer to our method. Yet, we consistently outperform all of them on all datasets. In one case (namely, on Paris 106k), our method is more than 14 mAP points ahead of the best competitor (Radenovic et al. 2016).

The de facto evaluation protocol for methods based on CNN features on the Holidays dataset involves manually rotating the images to correct their orientation. If we do not manually rotate the images, our accuracy drops from 94.8 to 90.3, which still outperforms the current state of the art. Instead of using an oracle to rotate the database images, one can automatically rotate the query image and issue three different queries (original query, query rotated 90 degrees, and query rotated 270 degrees). The score of one database image is the maximum score obtained with the three queries. This makes the query process 3 times slower, but improves the accuracy to 92.9 with no oracle intervention.

We also include our reimplementation of the R-MAC baseline (Tolias et al. 2016) using ResNet101 instead of VGG16. Although the accuracy improvement when using ResNet101 is not negligible, the accuracy obtained by the trained model is still much higher (in Oxford, 69.4 without training vs 84.1 and 86.1 when training, either using single-resolution or multi-resolution setting). This gap underlines the importance of both a well designed architecture and a sound end-to-end training with relevant data, all tailored to the particular task of image retrieval.

The second part of Table 4 shows results for state-of-the-art methods that do not necessarily rely on a global representation. The majority of them is characterized by a larger memory footprint than our method, *e.g.* the ones of Tolias and Jégou (2015), Tolias et al. (2016), Danfeng et al.

Table 4 Accuracy comparison with the state of the art

Method	Dim.	Datasets				
		Oxf5k	Par6k	Oxf105k	Par106k	Holidays
Global descriptors						
Jégou and Zisserman (2014)	1024	56.0	–	50.2	–	72.0
Jégou and Zisserman (2014)	128	43.3	–	35.3	–	61.7
Gordo et al. (2012)	512	–	–	–	–	79.0
Babenko et al. (2014)	128	55.7*	–	52.3*	–	75.9/78.9$^{\triangleright}$
Gong et al. (2014)	2048	–	–	–	–	80.8
Babenko and Lempitsky (2015)	256	53.1	–	50.1	–	80.2$^{\triangleright}$
Ng et al. (2015)	128	59.3*	59.0*	–	–	83.6
Paulin et al. (2015)	256k	56.5	–	–	–	79.3
Perronnin and Larlus (2015)	4000	–	–	–	–	84.7
Tolias et al. (2016)	512	66.9	83.0	61.6	75.7	85.2†/86.9$^{\dagger,\triangleright}$
Tolias et al. (2016) (ResNet101)†	2048	69.4	85.2	63.7	77.8	91.3 $^{\triangleright}$
Kalantidis et al. (2016)	512	68.2	79.7	63.3	71.0	84.9
Arandjelovic et al. (2016)	4096	71.6	79.7	–	–	83.1/87.5$^{\triangleright}$
Radenovic et al. (2016)	512	79.7	83.8	73.9	76.4	82.5$^{\triangleright}$
Previous state of the art		79.7 Radenovic et al. (2016)	83.8 Radenovic et al. (2016)	73.9 Radenovic et al. (2016)	76.4 Radenovic et al. (2016)	84.9 Kalantidis et al. (2016)
Ours	2048	**86.1**	**94.5**	**82.8**	**90.6**	90.3/**94.8**$^{\triangleright}$
Matching/Spatial verification/QE						
Chum et al. (2011)		82.7	80.5	76.7	71.0	–
Danfeng et al. (2011)		81.4	80.3	76.7	–	–
Mikulik et al. (2013)		84.9	82.4	79.5	77.3	75.8$^{\triangleright}$
Shen et al. (2014)		75.2	74.1	72.9	–	76.2
Tao et al. (2014)		77.8	–	–	–	78.7
Deng et al. (2013)		84.3	83.4	80.2	–	84.7
Tolias et al. (2015)		86.9	85.1	85.3	–	81.3
Tolias et al. (2016)	512	77.3	86.5	73.2	79.8	–
Tolias et al. (2016) (ResNet101)†	2048	78.9	89.7	75.5	85.3	
Tolias and Jégou (2015)		89.4	82.8	84.0	–	–
Li et al. (2015)		73.7	–	–	–	89.2
Kalantidis et al. (2016)	512	72.2	85.5	67.8	79.7	–
Radenovic et al. (2016)	512	85.0	86.5	81.8	78.8	–
Azizpour et al. (2015)		79.0	85.1	–	–	**90.0**
Previous state of the art		89.4 Tolias and Jégou (2015)	86.5 Tolias et al. (2016)	85.3 Tolias et al. (2015)	79.8 Tolias et al. (2016)	**90.0** Azizpour et al. (2015)
Ours (with QE)	2048	90.6	96.0	89.4	93.2	–
Ours (with QE and DBA)	2048	**94.7**	**96.6**	**93.6**	**93.5**	–

Bold values indicate the best results per dataset for each type of methods
Methods marked with an * use the full image as a query in Oxford and Paris instead of using the annotated region of interest as is standard practice. The † symbol denotes our reimplementation. Methods that manually rotate the images on Holidays using an oracle are labeled with $\triangleright$. We do not perform QE on Holidays as it is not a standard practice. See text for more details

(2011), Azizpour et al. (2015). These methods perform a costly spatial verification at runtime that typically requires storing thousands of local descriptors for each image in the database (Tolias and Jégou 2015; Li et al. 2015; Mikulik et al. 2013). Most of them also perform query expansion (QE). For comparison purposes, we also report our results using

QE with or without DBA at the bottom of the table. Using only QE brings about half of the improvement obtained when using both QE and DBA, yet avoiding any pre-processing of the database. In spite of not requiring any form of spatial verification at runtime, our method is largely improving on the state of the art on all datasets. In particular, our performance is between 5 to 14 mAP points ahead of the best competitor on all datasets.

The best methods in the literature (Tolias and Jégou 2015; Azizpour et al. 2015) are hardly scalable as they require a lot of storage memory and an expensive verification. For instance, the method of Tolias and Jégou (2015) requires a slow spatial verification taking over 1 second per query (excluding descriptor extraction time). Without spatial verification their approach loses 5 mAP points and still requires about 200 ms per query. The approach of Tolias et al. (2016) is more scalable but still needs an extra spatial-verification stage based on storing many local representations of the database images, ending up in a significantly larger memory footprint than our approach, despite using advanced compression techniques. In comparison, our approach only calculates two matrix-vector products (only one if QE is not performed), that are extremely efficient. This operation computes several millions of image comparisons in less than a second. Without any compression, our method requires storing 2048 floats per image, *i.e.* 8 kb, but this representation can be drastically compressed without much accuracy loss as we show in the next section. Finally, we would like to point out that our method uses a single universal model to compute our learned image representation—the same for all test datasets—contrary to, for instance, other methods of Danfeng et al. (2011), Shen et al. (2014), Tolias et al. (2015) that perform some learning on the target datasets.

We also report results on the UKB dataset using our universal model. Our method obtains 3.84 recall@4 without QE and DBA, and 3.91 recall@4 score with QE and DBA. The latter is comparable to the best published results on this dataset, *i.e.* 3.85 reported by Azizpour et al. (2015), although this method is a lot more costly. Other results are significantly lower (*e.g.* Paulin et al. (2015) reports 3.76, Deng et al. (2013) reports 3.75, and Tolias and Jégou (2015) reports 3.67) and they are hardly scalable as well (see discussion above). Note that training marginally decreases our performance on UKB (Table 1). This is caused by the discrepancy between our training set (landmarks images) and the UKB images (daily life items). The drop remains marginal, which suggests that out method adapts well to other retrieval contexts.

6.3.3 Short Image Codes with PCA and PQ

We investigate two different methods to reduce the memory footprint of our approach while preserving the best possible accuracy. We compress our 2048-dimensional image descriptors using either principal component analysis (PCA) or product quantization (PQ) (Jégou et al. 2011). In both cases, we learn the vocabulary (PCA projection or PQ codebook) on Landmarks-clean images, encoded with our learned representation.

In the case of PCA, to obtain descriptors of d dimensions we simply mean center the features, project them with the eigenvectors associated with the d largest eigenvalues of the data, and ℓ_2-normalize them. The resulting descriptor size is thus $4d$ bytes, as they are stored as 32-bits floats. PQ compression, for its part, is based on splitting the input descriptor in k subparts and applying vector quantization on each subpart separately. Although some works also apply PCA to the input descriptors before the PQ encoding, we found it did not have any noticeable impact in our case. Training PQ is then equivalent to learning a codebook for each subpart and is achieved though k-means clustering on a set of representative descriptors. The codebook size is typically set to 256 for each subpart, as it allows them to be stored on exactly 1 byte. Thus, the size of a PQ-encoded descriptor is k bytes. At test time, efficient caching techniques allow to compute the dot-product between the query and the PQ-encoded database descriptors efficiently (Jégou et al. 2011). Note that recent improvements have led PQ to match the high speed of bitwise Hamming distance computations without losing in accuracy (Douze et al. 2016).

Retrieval results for our method (without QE or DBA) and for the state of the art are presented in Fig. 8 for all datasets and for different descriptor sizes (in bytes). PCA-based compression, labeled as "Proposed (PCA)", achieves slightly better results than other existing approaches for all considered datasets and all code sizes, but its accuracy drops rapidly for short codes. This compression method is still of interest as it does not require any change in the system architecture and still compares favorably to the state of the art. PQ-based compression, labeled as "Proposed (PQ)" in Fig. 8, largely outperforms all published methods in terms of the performance versus size trade-off by a large margin, on all datasets. Even for very short image codes of 64 bytes, it is able to outperform most of the state of the art that uses codes of 2048 bytes. In this setting, we can store hundreds of millions of images on a single machine with 64 Gb of RAM, which demonstrates the scalability of our approach.

6.4 Qualitative Results

Figure 9 shows the top retrieved images by our final best performing retrieval system based on ResNet101 (including QE and DBA) on some Oxford 5k queries (purple rectangle on the leftmost images). For every query we also provide the corresponding average precision (AP) curve (green curve) and compare it with the ones obtained for the baseline R-MAC

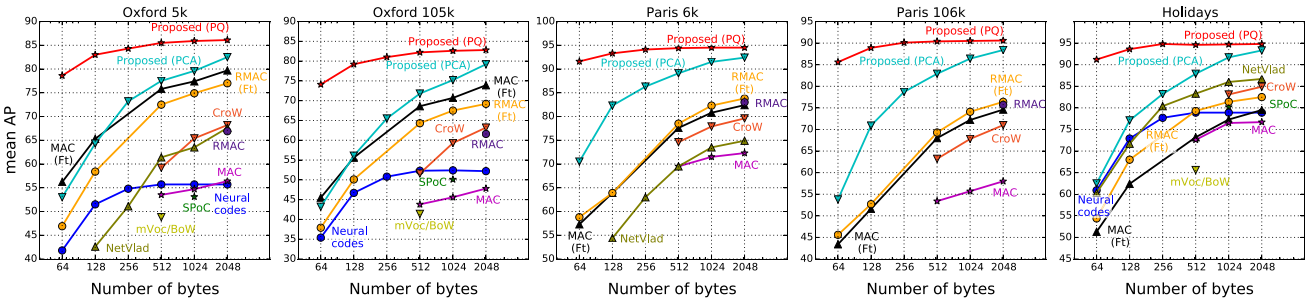

Fig. 8 Results for short image codes. Our method with PQ and PCA compression, compared to finetuned MAC and R-MAC (Radenovic et al. 2016), CroW (Kalantidis et al. 2016), MAC and R-MAC (Tolias et al. 2016), Neural codes (Babenko et al. 2014), NetVlad (Arandjelovic et al. 2016), SPoC (Gong et al. 2014), and mVOC/BoW (Radenovic et al. 2015)

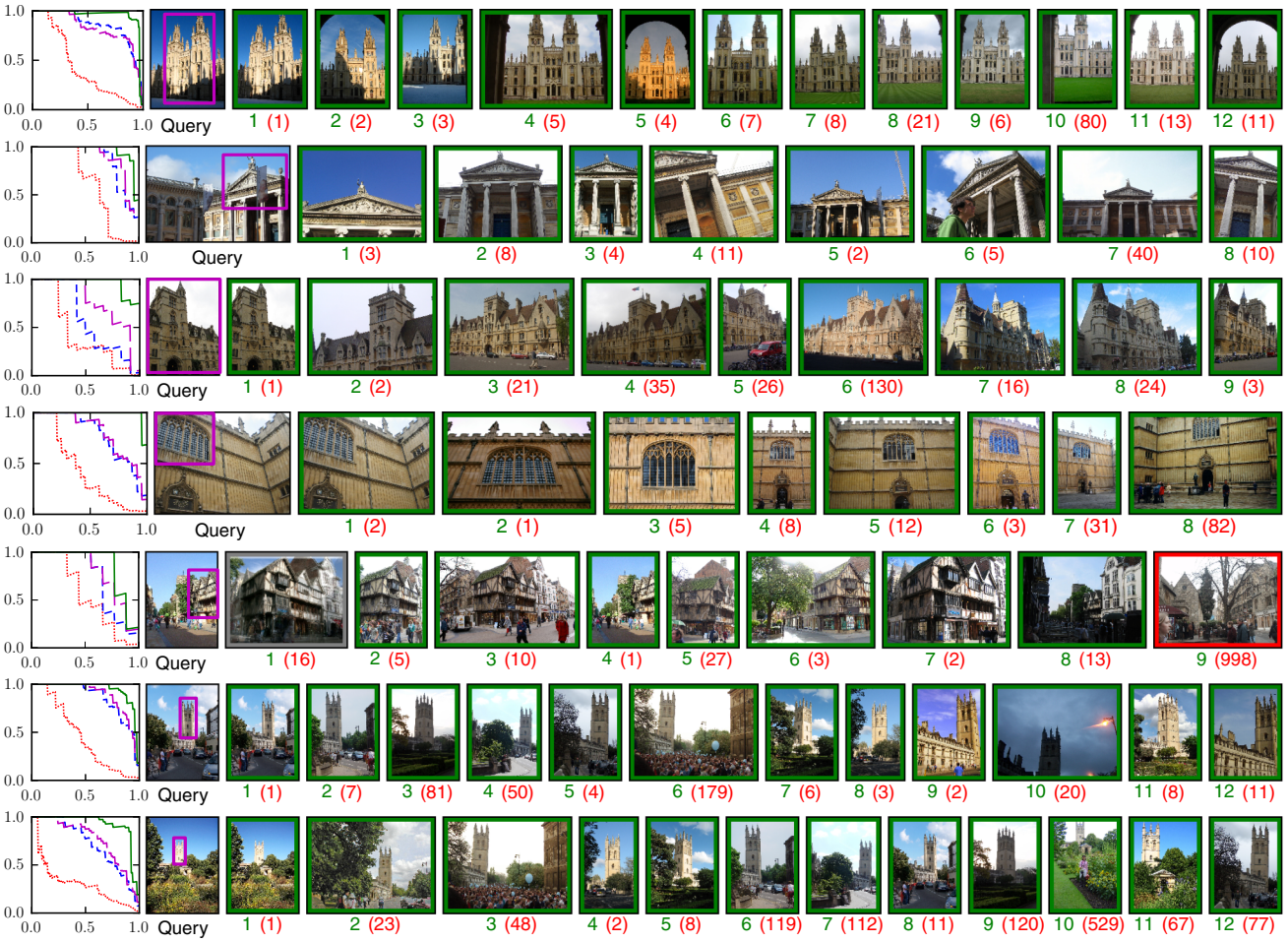

Fig. 9 *Top* retrieved images and AP curves for a few Oxford queries (*purple rectangle* from leftmost images). On the plot, R-MAC baseline is *red*, our learned version is *blue*, multi-resolution is *purple*, and the full system with QE and DBA is *green*. *Green*, *gray* and *red borders* on images respectively denote positive, null and negative images. Below each image is its retrieval rank, using the same colour code (Color figure online)

(red curve), our learned architecture (blue curve), and its multi-resolution flavor (purple curve). The results obtained with the proposed trained model are consistently better in terms of accuracy. In many cases, several of the correctly retrieved images by our method were not well scored by the baseline method, that placed them far down in the list of results.

7 Conclusions

We have presented an effective and scalable method for instance-level image retrieval that encodes images into compact global signatures that can be compared with the dot-product. The proposed approach combines three ingredients that are key to success. First, we gathered a suitable training set by automatically cleaning an existing landmarks dataset. Second, we proposed a learning framework that relies on a triplet-based ranking loss, and that leverages this training set to train a deep architecture. Third, for the deep architecture, we built on the R-MAC descriptor, cast it as a fully differentiable network so we could learn its weights, and enhanced it with a proposal network that focuses on the most relevant image regions. Extensive experiments on several benchmarks show that our representation significantly outperforms the state of the art when using global signatures, even when using short codes of 64 or 128 bytes. Our method also outperforms the state of the art set by more complex methods that rely on costly matching and verification, and does so while being faster and more memory-efficient.

References

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., et al. (2015). Vqa: Visual question answering. In *ICCV*.

Arandjelovic, R., & Zisserman, A. (2012). Three things everyone should know to improve object retrieval. In *CVPR*.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*.

Azizpour, H., Razavian, A., Sullivan, J., Maki, A., & Carlsson, S. (2015). Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (99):1–1.

Babenko, A., & Lempitsky, V. S. (2015). Aggregating deep convolutional features for image retrieval. In *ICCV*.

Babenko, A., Slesarev, A., Chigorin, A., & Lempitsky, V. S. (2014). Neural codes for image retrieval. In *ECCV*.

Chopra, S., Hadsell, R., & Lecun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of computer vision and pattern recognition conference*.

Chum, O., Philbin, J., Sivic, J., Isard, M., & Zisserman, A. (2007). Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*.

Chum, O., Mikulik, A., Perdoch, M., & Matas, J. (2011). Total recall II: Query expansion revisited. In *CVPR*.

Danfeng, Q., Gammeter, S., Bossard, L., Quack, T., & Van Gool, L. (2011). Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *CVPR*.

Deng, C., Ji, R., Liu, W., Tao, D., & Gao, X. (2013). Visual reranking through weakly supervised multi-graph learning. In *ICCV*.

Deng, J., Dong, W., Socher, R., Li, LJ., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *CVPR*.

Douze, M., Jegou, H., & Perronnin, F. (2016). Polysemous codes. In *ECCV*.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., & Mikolov, T. (2013). Devise: A deep visual-semantic embedding model. In *NIPS*.

Girshick, R. (2015). Fast R-CNN. In *CVPR*.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*.

Gordo, A., Rodríguez-Serrano, J. A., Perronnin, F., & Valveny, E. (2012). Leveraging category-level labels for instance-level image retrieval. In *CVPR*.

Gordo, A., Almazán, J., Revaud, J., & Larlus, D. (2016). Deep image retrieval: Learning global representations for image search. In *ECCV*.

Hadsell, R., Chopra, S., & Lecun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *CVPR*.

Hays, J., & Efros, A. A. (2008). im2gps: Estimating geographic information from a single image. In *CVPR*.

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. In *SIMBAD*.

Hu, J., Lu, J., & Tan, Y. P. (2014). Discriminative deep metric learning for face verification in the wild. In *CVPR*.

Jégou, H., & Chum, O. (2012). Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening. In *ECCV*.

Jégou, H., & Zisserman, A. (2014). Triangulation embedding and democratic aggregation for image search. In *CVPR*.

Jégou, H., Douze, M., & Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*.

Jégou, H., Douze, M., & Schmid, C. (2010). Improving bag-of-features for large scale image search. In *IJCV*.

Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR*.

Jegou, H., Douze, M., & Schmid, C. (2011). Product quantization for nearest neighbor search. In *TPAMI*.

Kalantidis, Y., Mellina, C., & Osindero, S. (2016). Cross-dimensional weighting for aggregated deep convolutional features. In *Workshop on web-scale vision and social media (VSM), ECCV*.

Karpathy, A., Joulin, A., & Fei-Fei, L. (2014). Deep fragment embeddings for bidirectional image-sentence mapping. In *NIPS*.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *NIPS*.

Laptev, D., Savinov, N., Buhmann, J. M., & Pollefeys, M. (2016). Ti-pooling: Transformation-invariant pooling for feature learning in convolutional neural networks. In *CVPR*.

Li, X., Larson, M., & Hanjalic, A. (2015). Pairwise geometric matching for large-scale object retrieval. In *CVPR*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. In *IJCV*.

Makadia, A., Pavlovic, V., & Kumar, S. (2008). A new baseline for image annotation. In *ECCV*.

Mikolajczyk, K., & Schmid, C. (2004), Scale and affine invariant interest point detectors. In *IJCV*.

Mikulík, A., Perdoch, M., Chum, O., & Matas, J. (2010). Learning a fine vocabulary. In *ECCV*.

Mikulik, A., Perdoch, M., Chum, O., & Matas, J. (2013). Learning vocabularies over a fine quantization. In *IJCV*.

Ng, J. Y. H., Yang, F., & Davis, L. S. (2015). Exploiting local features from deep networks for image retrieval. In *CVPR workshops*.

Nister, D., & Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*.

Paulin, M., Douze, M., Harchaoui, Z., Mairal, J., Perronin, F., & Schmid, C. (2015). Local convolutional features with unsupervised training for image retrieval. In *ICCV*.
Perdoch, M., Chum, O., & Matas, J. (2009). Efficient representation of local geometry for large scale object retrieval. In *CVPR*.
Perronnin, F., & Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *CVPR*.
Perronnin, F., & Larlus, D. (2015). Fisher vectors meet neural networks: A hybrid classification architecture. In *CVPR*.
Perronnin, F., Liu, Y., Sánchez, J., & Poirier, H. (2010). Large-scale image retrieval with compressed fisher vectors. In *CVPR*.
Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *CVPR*.
Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*.
Philbin, J., Isard, M., Sivic, J., & Zisserman, A. (2010). Descriptor learning for efficient retrieval. In *ECCV*.
Radenovic, F., Jegou, H., & Chum, O. (2015). Multiple measurements and joint dimensionality reduction for large scale image search with short vectors-extended version. In *International Conference on Multimedia Retrieval*.
Radenovic, F., Tolias, G., & Chum, O. (2016). CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *ECCV*.
Razavian, A.S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR deep vision workshop*.
Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.
Rodriguez-Serrano, J., Larlus, D., & Dai, Z. (2015). Data-driven detection of prominent objects. In *TPAMI*.
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, AC., & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. In *IJCV*.
Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *CVPR*.
Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *CVPR*.
Shen, X., Lin, Z., Brandt, J., & Wu, Y. (2014). Spatially-constrained similarity measurefor large-scale object retrieval. In *TPAMI*.
Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., & Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. In *ICCV*.
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.
Sivic, J., & Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *ICCV*.
Song, H.O., Xiang, Y., Jegelka, S., & Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *CVPR*.
Sun, Y., Chen, Y., Wang, X., & Tang, X. (2014). Deep learning face representation by joint identification-verification. In *NIPS*.
Tao, R., Gavves, E., Snoek, C.G., & Smeulders, AW. (2014). Locality in generic instance search from one example. In *CVPR*.
Tolias, G., & Jégou, H. (2015). Visual query expansion with or without geometry: Refining local descriptors by feature aggregation. In *PR*.
Tolias, G., Avrithis, Y., & Jégou, H. (2015). Image search with selective match kernels: Aggregation across single and multiple images. In *IJCV*.
Tolias, G., Sicre, R., & Jégou, H. (2016). Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*.
Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on PAMI, 30*(11), 1958–1970. doi:10.1109/TPAMI.2008.128.
Turcot, P., & Lowe, D.G. (2009). Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshops*.
Vardi, Y., & Zhang, C. H. (2004). The multivariate L1-median and associated data depth. In *Proceedings of the National Academy of Sciences*.
Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., et al. (2014) Learning fine-grained image similarity with deep ranking. In *CVPR*.