Machine Vision and Applications (2022) 33:16
https://doi.org/10.1007/s00138-021-01260-z

SPECIAL ISSUE PAPER

Effective triplet mining improves training of multi-scale pooled CNN for image retrieval

Federico Vaccaro[1] · Marco Bertini[1] · Tiberio Uricchio[1] · Alberto Del Bimbo[1]

Received: 14 March 2021 / Revised: 19 August 2021 / Accepted: 24 October 2021

Abstract
In this paper, we address the problem of content-based image retrieval (CBIR) by learning images representations based on the activations of a Convolutional Neural Network. We propose an end-to-end trainable network architecture that exploits a novel multi-scale local pooling based on the trainable aggregation layer NetVLAD (Arandjelovic et al in Proceedings of the IEEE conference on computer vision and pattern recognition CVPR, NetVLAD, 2016) and bags of local features obtained by splitting the activations, allowing to reduce the dimensionality of the descriptor and to increase the performance of retrieval. Training is performed using an improved triplet mining procedure that selects samples based on their difficulty to obtain an effective image representation, reducing the risk of overfitting and loss of generalization. Extensive experiments show that our approach, that can be effectively used with different CNN architectures, obtains state-of-the-art results on standard and challenging CBIR datasets.

Keywords Image retrieval · CBIR · CNN · NetVLAD · Max-pooling

1 Introduction

Content-based image retrieval (CBIR) has received large attention both from computer vision and multimedia scientific communities since the early 1990s. In many contexts an image has to be retrieved using another image, by the objects depicted, the style of the image or its purpose. The inception of social networks has further increased the need of CBIR techniques to manage the ever copious number of images produced every minute in the world [6,24]. At the end of the "early years" of image retrieval [48] global and local visual cues such as texture, color and shape were commonly used to index images. Then, for about 10 years, approaches based on local invariant features like SIFT and representations based on Bag-of-Words and its variants [e.g. Fisher Vectors and VLAD (Vector of Locally Aggregated Descriptors [9])] have obtained state-of-the-art results. Nowadays, since the inception of Convolutional Neural Networks (CNNs), approaches using either convolutional or fully connected layer activations have started to obtain better results [40] than those that aggregate local manually engineered features. The most recent CNN-based approaches aggregate regional activations, learning image representations in an end-to-end approach [39], somewhat fulfilling the forecast made 20 years ago in [48]. "Learning is quickly gaining attention as a means to build explicit models for each semantic term. … It is our view that, in order to bring semantics to the user, learning is inevitable".

We start from two complimentary observations. On the one hand, global descriptors are robust to appearance and illumination changes since they are directly optimized to recognize places [57]. On the other hand, local descriptors, thanks to their fixed spatial neighbourhood, have high spatial precision and provide highly accurate pose estimation [12], also requiring aggregation [57] sometimes at the expense of high dimensionality [1]. Few works have attempted to develop a combination of them [7,44,50]. Our novel model takes the strengths of global and local features, while reducing their weaknesses. Hence, in this paper, we present a

✉ Tiberio Uricchio
tiberio.uricchio@unifi.it

Federico Vaccaro
federico.vaccaro@unifi.it

Marco Bertini
marco.bertini@unifi.it

Alberto Del Bimbo
albertoDel.bimbo@unifi.it

[1] Media Integration and Communication Center (MICC), University of Florence, Viale Morgagni, 65, Florence, Italy

Published online: 06 January 2022

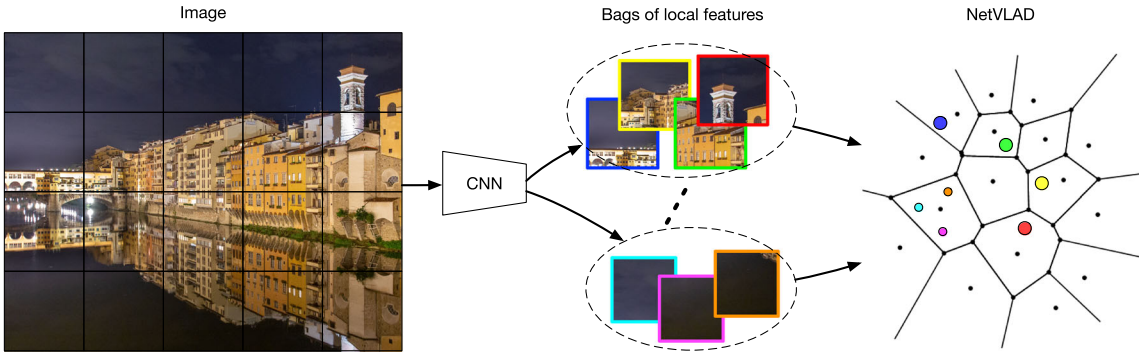

Fig. 1 The basic idea of our proposed method: an image is split into bags of local features which are weighted and pooled with NetVLAD. The entire process is automatically learned with a deep neural network trained with a triplet mining procedure

novel multi-scale CNN regions pooling that aggregates local features before performing a second aggregation step using NetVLAD [1]. NetVLAD, a generalized form of the hand-crafted VLAD, is an end-to-end trainable layer for CNN that performs aggregation of descriptors, currently considered state of the art aggregator for local descriptors in recent applications [16,52]. To reduce NetVLAD dimensionality the features are split in bags of local features. This is used in an end-to-end learning approach in conjunction with a 3-stream Siamese network, to learn optimized image representations. A second contribution of this work is an improved triplet mining procedure, with a complexity of $O(n)$, that provides a diverse set of semi-hard and hard triplets, avoiding extremely hard ones that may hinder learning and avoiding excessively similar triplets that may lead to overfitting. The proposed method is evaluated on two standard image retrieval datasets: Revisited Oxford5K and Revisited Paris6K, as well as the previous versions Oxford5K and Paris6K, outperforming previous state-of-the-art results. Experiments show that the proposed approach can be applied to different network architectures, from VGG [46] and ResNet [17], used to provide a fair comparison with competing approaches, to the very recent ResNeSt [58], used to show the applicability to more modern computationally efficient backbones.

The paper is organised as follows: Discussion of previous works is provided in Sect. 2; description of the proposed method and its contributions is provided in Sect. 3; experiments on standard CBIR datasets to evaluate the proposed contributions and a thorough comparison with competing state-of-the-art approaches are reported in Sect. 4; finally, conclusions are drawn in Sect. 5.

2 Previous works

After the introduction of the successful Bag-of-Visual-Words model in [47], many works have improved over it addressing aspects such as approximating local descriptors [21], learning improved codebooks [27], improving local features aggregation [9,20,34]. In the last years, following the success obtained using CNNs to solve the problem of image classification, CNN-based features have started to be used also for image retrieval tasks. A thorough survey that compares SIFT-based and CNN-based approaches for instance-based image retrieval is provided in [59].

2.1 CNN-based feature extraction

The most basic CNN-based approach is to use the convolutional network as a feature extractor. That is achieved by taking the activations of fully connected or convolutional layers as descriptors. The use of the first fully connected layer of AlexNet (FC6) has been used in [40], outperforming local features approaches for instance retrieval in several standard datasets. The performance of different AlexNet layers and the effects of PCA have been evaluated in [5]. The relative performance of the aggregation methods for deep convolutional features was shown to be different than the case of shallow descriptors [4]. More recent approaches have used max-pooled or sum pooling activations from convolutional layers as in [2,4,41,60].

CNN features can be aggregated using the Bag-of-Words model applied to local convolutional features as in [29]; VLAD has been applied to global features as in [57] and to local patches as in [12,57]; Fisher Vectors have been applied to localized local feature maps derived from objectness detectors as in [50,57]. Component-wise max-pooling of CNN features computed on object proposals has been used in [42]. The application of these approaches to compute CNN-based features may have a negative impact on the computational performance: the approaches used in [12,40] require to compute CNN features on a large number of sub-patches, an operation that may not be practical when dealing with large datasets; this problem has been addressed in [50,57], where object proposals and "dense sampling" from max-pooling of convolutional layers are used. At the same time, aggregations

were also studied in [4], showing that performing classical aggregation, after learning a CNN, can obtains inferior performance compared to even simple sum pooling present at training time.

Another solution is to use faster pooling approaches. Regional maximum activation of convolutions (R-MAC) aggregation [49] consider a set of fixed squared regions at different scales, and collects the maximum response in each channel. These descriptors are sum-pooled to create the final R-MAC descriptor. Hashing of CNN features, either global [11,30] or local, based on objectness scores [54], have been used to speed-up image retrieval tasks.

2.2 End-to-end approaches

In this class of approaches, CNN models are fine-tuned on a training set to learn better representations or aggregations of features, so to extract features through a single pass of the model and learning them in an end-to-end manner. Typically this results in an improved performance w.r.t. methods based on CNN feature extraction only [13,38].

In [1] has been proposed a layer called NetVLAD, pluggable in any CNN architecture and trainable through back-propagation, that is inspired by the commonly used VLAD representation. This allows to train end-to-end a network using an aggregation of VGG16 convolutional activations. Multi-scale pooling of CNN features followed by NetVLAD has been proposed in [51], obtaining state-of-the-art results using VGG16. Simultaneous learning of CNN and Fisher Vector parameters using a Siamese network and contrastive loss has been proposed in [32].

The current state-of-the-art approaches such as [14,39,43,51] follow an end-to-end approach: [14] uses a three-stream Siamese network with triplet loss, [39] uses a two-stream Siamese network with contrastive loss, and [43] directly optimizes for the mean Average Precision metric using thousands of images instead of pairs or triplets. In [3] is proposed a so-called SPoC descriptor, based on the aggregation of raw deep convolutional features without embedding. In [14] an end-to-end learned version of R-MAC descriptor is presented, along with a triplet mining procedure to efficiently train a three streams Siamese Network using triplet loss. In this approach, a region proposal network selects the most relevant regions of the image, where local features are extracted, in three scales of the input images. In [39] a trainable Generalized-Mean (GeM) pooling layer is proposed, along with learning whitening, for short representations. Two stream Siamese network is trained using contrastive loss. The authors use structure-from-motion information and hard-matching examples for CNN training, and use up to 5 image scales to extract features. More recently, in [18] a method that uses a novel conditional attention model to localize region of interests is proposed, combining it with a GeM pooling layer. In [43] it is proposed to directly optimize the global mAP score by leveraging list-wise loss formulation, using a histogram binning approximation that can be employed for end-to-end learning. In this approach it is possible to use large batches of high-resolution images during training and reduce the need of engineering mining procedures. In [7] an end-to-end trainable "DEep Local and Global features" (DELG) model, that combines global features for first stage retrieval and local features for re-ranking, has been recently proposed. It uses GeM pooling for global features and attentive selection for local features. In [31] it is proposed to exploit the second-order relationships between features at different spatial locations; in this approach feature maps are re-weighted according to second-order spatial attention, improving results on challenging datasets.

In [25] a novel spatial attention block to weight salient regions is combined with semantic segmentation and a class weighting network to implement an end-to-end semantic-aware object retrieval based on R-MAC. In [45] is presented a method called Strong-Response-Stack-Contribution (SRSC), that computes a global representation vector combining spatial and channel contribution of VGG features.

In [10] vision transformers are proposed to generate image descriptors for retrieval, training the models with a metric learning objective that combines a contrastive loss with a differential entropy regularizer. In [26], the problem of retrieval under noisy datasets is addressed proposing a novel noise-robust loss based on Multiple Instance Learning (MIL). The proposed method allows to use noisy generated training sets, that can be easily created, to adapt CNNs for image retrieval on new objects. In [61] a novel metric learning approach is proposed, that shifts part of the learning to an online local metric adaptation. The goal is to reduce the demand of a huge set of positive/negative training pairs, and in particular to address the problem of obtaining positive pairs for a specific instance.

Our proposed method shares similarity with the approaches of [14,39,51], but in addition to our proposed pooling and triplet mining, it has important subtle differences that increase performance of the resulting system, compared to previous methods [7,43]. Differently from [12,50,57] our method is fully trainable end-to-end; differently from [57] multiple scales and only one convolutional layer are used; differently from [12] the VLAD aggregation is performed concurrently at all the scales, and differently from [50] there is no use of region proposals. Differently from [1], our input to the NetVLAD layer is not directly convolutional activations, but the concatenation of two max-pooled sets of activations. Differently from [51] we add a ReLU layer before multi-scale pooling, the feature splitting, a PCA that allows to greatly reduce the dimensionality of the embedding and a new triplet mining procedure.

3 The proposed method

The idea is to train a CNN network which provides optimized embeddings to perform image retrieval.[1] The proposed method is inspired by the approaches used in [1,14,39,51], with several differences. First, the CNN activations are collected using three different aggregation steps. The first one through max-pooling operations, i.e. using 2-scales local features; the second through an operation of 'splitting' the high-dimensional features into a more compact descriptor with reduced curse of dimensionality. A bag of words representation is obtained and fed into the NetVLAD layer. A second difference is the triplet (*hard-negatives*) mining procedure used to train a three-stream Siamese network. We select "stable" triplets, whose loss is targeted around a given value, avoiding positive samples that could be considered as extremely hard, i.e. whose visual similarity is very low due to minimal overlap, extreme zooming [39], etc. This procedure reduces the risk of overfitting and loss of generalization, also due to mathematical implications related to the loss gradients [53].

3.1 Pooling of local CNN features and descriptor splitting

Convolutional features are max-pooled using a 2×2 and 3×3 (both using stride $= 1$) process, so to obtain representations at finer and larger detail where each local feature encodes information about its neighbourhood. This operation produces two batches of local descriptors. For each location of the two partitions the f activation maps are collected, creating a $1 \times 1 \times f$ "column feature" (as defined in [59]). This process, shown in Fig. 2, is akin to dense grid-based sampling of SIFT descriptors [19]. Sets of column features are concatenated, to provide a multi-scale descriptor of the image. After this step, we perform *feature splitting*, that is the operation of splitting each "column feature" into a given number of equal dimensional features. Given a D-dimensional vector $\mathbf{f}$, with $D \mod N = 0$:

$$\mathbf{f} = [f_0, f_1, \ldots, f_i, f_{i+1}, \ldots, f_{D-1}] \quad (1)$$

We refer to the *N-Split* of $\mathbf{f}$ to the set:

$$\mathbf{f^N} = \{\mathbf{f_i} = [f_{i \times D/N}, \ldots, f_{(i+1) \times D/N}], i \in [0, N-1]\} \quad (2)$$

This splitting is simply implemented by a *tensorial reshaping* and each N-Split is obtained starting from the features contained in the bag of words. The purpose of this transformation is reducing the dimensionality of the embeddings into a more suitable form for the NetVLAD layer, and avoiding the dimensionality explosion of the final VLAD descriptor that would happen combining non-trivial vocabulary size with highly-dimensional features, like the features produced by modern network architectures (e.g. ResNet has 2048 filters in the output layer).

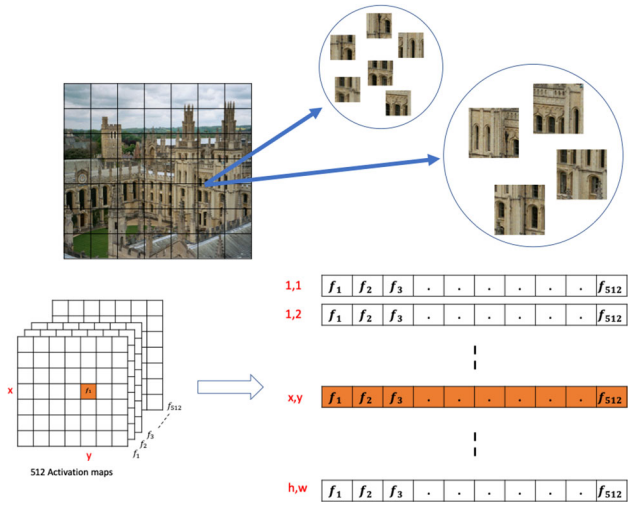

Fig. 2 "Column feature" extraction: top) max-pooling with different scales, bottom) activation maps collection as column features: this is performed at each pooling scale

All the local CNN features are then aggregated using a NetVLAD [1] layer. The activations of this layer are used as a descriptor of the content of the image. The NetVLAD layer is initialized with a K-Means clustering,[2] with the soft-assignment parameter $\alpha = 30$. As in [1] for NetVLAD we use $K = 64$ vocabulary words; we used for the ResNet architecture the N-Split of the local features with $N = 4$, thus resulting in a 32k-D representation.

The approach can be applied in principle to any CNN. In the following experiments we have tested VGG16, as it was commonly used in many competing methods and comparisons, ResNet, as it has been used in the most recent state-of-the-art methods, and the novel ResNeSt architecture [58] that has recently improved results over previous ResNet in image classification, object detection, instance segmentation and semantic segmentation.

An overview of the proposed network is shown in Fig. 3. The FCNN block in figure is the CNN used to extract features; we extract the feature from the last convolutional layer of each tested network. The FCNN block is followed by a ReLU block, followed by the two local max-pooling aggregations. The bag of local features that are aggregated by the NetVLAD layer are obtained by the union of the feature splitting of the column features of the two max-pooling layers.

[1] The code and models are released at the following address: https://github.com/fede-vaccaro/NetVlad.

[2] In the experiments we performed it on the training dataset.

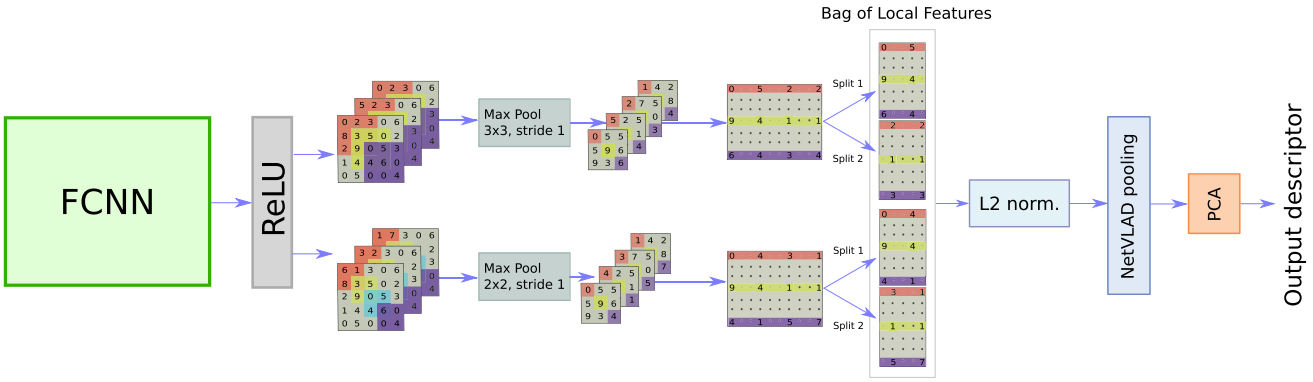

Fig. 3 Overview of the proposed architecture, using VLAD aggregation of local multiscale max-pooling CNN features

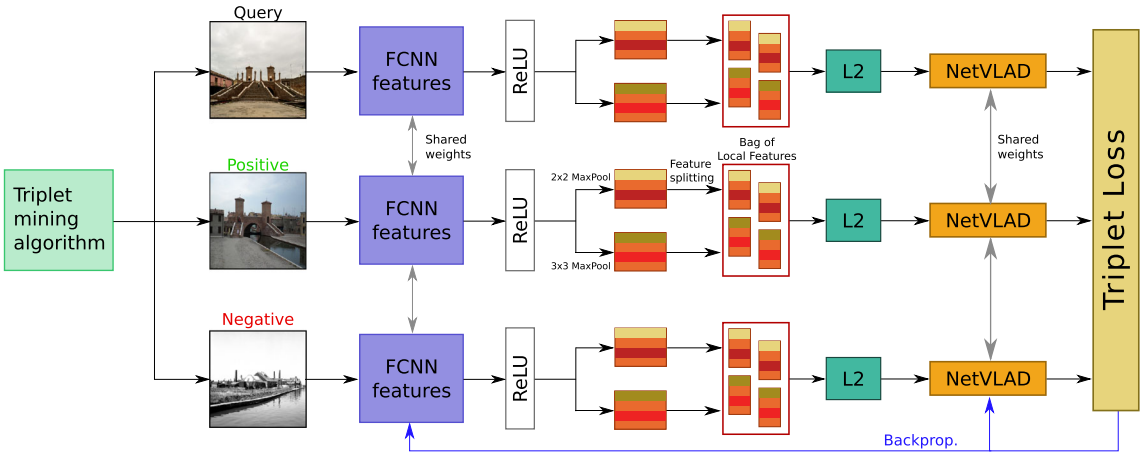

Fig. 4 Schema of the proposed training method: the three stream Siamese network is used at training time. At test time the query image is fed to the learned network to produce an effective image representation used to query the database

3.2 Training and triplet mining

In this work we use a ranking loss based on triplets of images; the idea is to learn a descriptor so that the representation of relevant images is closer to the descriptor of the query than that of irrelevant ones. The design of the training setup is shown in Fig. 4: the weights of the convolutional layers of the CNN network and the NetVLAD layer are shared between the streams, since their size is independent of the size and aspect ratio of the images.

At training time the network is provided with image triplets. Given a query image Q with embedding $\mathbf{q}$, a positive (i.e. relevant with respect to the query) image P with embedding $\mathbf{p}$, a negative (i.e. not relevant w.r.t. the query) image N with embedding $\mathbf{n}$, with $\mathbf{q}, \mathbf{p}, \mathbf{n} \in \mathbb{R}^n$, a distance measure $d : \mathbb{R}^n \rightarrow \mathbb{R}$ (in our case it was the squared Euclidean distance) and a scalar m that controls the margin, the triplet loss used is $L = max(m + d(\mathbf{q}, \mathbf{p}) - d(\mathbf{q}, \mathbf{n}), 0)$. The margin m is set to 0.1 as in [1,13,39] which is proved to be a safe default value.

An issue that may arise with this approach is due to the sampling of the triplets: e.g. a random approach may select triplets that do not incur in any loss and thus do not improve the model. Before discussing Algorithm 1, we have to make considerations about the triplets and the samples from which they are composed, and their impact respect to the gradients and, therefore, the weight updates. For the sake of simplicity from here we refer to an image and its embedding indiscriminately. Given a distance measure d, a dataset of labeled images $\mathscr{D} = \{(\mathbf{x}, y)\}$ and a query image $(\mathbf{q}, y_q) \in \mathscr{D}$, we refer to the **hardest-positive** of $\mathbf{q}$, to the image $(\mathbf{p}, y_p)$ s.t.:

$$\mathbf{p} = \underset{\mathbf{p} \in \mathscr{D}}{argmax}\, d(\mathbf{q}, \mathbf{p}) \{\mathbf{p} | y_q = y_p\} \tag{3}$$

meaning the farther image from $\mathbf{q}$ belonging to the same class. Conversely, we refer to the **hardest-negative** of $\mathbf{q}$, the image $(\mathbf{n}, y_n)$ s.t.:

$$\mathbf{n} = \underset{\mathbf{n} \in \mathscr{D}}{argmin}\, d(\mathbf{q}, \mathbf{n}) \{\mathbf{n} | y_q \neq y_n\} \tag{4}$$

meaning the nearest image from $\mathbf{q}$ belonging to a different class. Ideally, for a perfect ranking, we would desire that $d(\mathbf{q}, \mathbf{p}) < d(\mathbf{q}, \mathbf{n})$ for each $\mathbf{q} \in \mathscr{D}$, so we are naively led to

Algorithm 1 Triplet mining

```
1: procedure TRIPLET MINING(mining_batch_size,
                images_per_classes, landmarks, t, T, δ)
2:     k ← mining_batch_size/images_per_classes
3:     Pick k random landmarks
4:     X, y ← pick mining_batch_size random images from
5:                          the selected landmarks and their labels
6:     features ← model.extract_features(X)
7:     triplets[] ← new list()
8:     for i ∈ [1, mining_batch_size] do
9:         if n ≠ null and j − n ≥ t then
10:            break
11:        end if
12:        feature = features[i]; query_label = y[i]
13:        indices[] ← Compute k-NN of feature
14:        q ← i; p ← null; n ← null
15:        for j ∈ [2, mining_batch_size] do
16:            if label[indices[j]] ≠ query_label and
                   n = null then
17:                n ← j
18:                if j > 2 with Probability 0.5 then
19:                    p ← j − 1
20:                    break
21:                end if
22:            else if label[indices[j]] = query_label and
                       n ≠ null then
23:                p ← j
24:            end if
25:            if p ≠ null and n ≠ null then
26:                loss_t ← compute_loss(X[q], X[p], X[n])
27:                if m − δ < loss_t < m + δ then
28:                    p ← indices[p]
29:                    n ← indices[n]
30:                    triplet ← (X[q], X[p], X[n])
31:                    triplets.append(triplet)
32:                else
33:                    break
34:                end if
35:            end if
36:        end for
37:    end for
38:    Keep just one triplet per class
39:    Keep at most T triplets
40:    return triplets
41: end procedure
```

sample **p** and **n** and train the model on triplets composed as this; however while it is common practice [13,39] performing the hard-negatives mining, selecting the hardest-positives involves two main issues:

– The gradient of the loss respect to the positive-term would have a large module; this implies that this gradient dominates the negative-term, thus the backpropagation is driven only from the positives samples, leading to a "collapsed" model: if the negative term is missing, the problem of minimizing the distance of a set of points would be trivially solved by mapping each image to a constant vector without any attention to keep the classes separated;

– Some classes may contain outliers and/or samples whose visual similarity is very low due to minimal overlap, extreme zooming [39]; images like these are more likely to be far from the query image. This information is encoded in the gradient via a noise-component although the distance may not be extremely large.

These phenomena have been analyzed in detail in [53]. The purpose of our mining algorithm is yielding hard-negative triplets, whose positive sample is selected s.t. is equally distant to the query w.r.t. the negative sample, implying that the loss at mining-time of these triplets is approximatively equal to the margin m and positive and negative gradients have equal strength. This avoids the positive collapse while keeping the positive samples non trivial.

Our algorithm operates by loading a subset $\mathscr{B} \subset \mathscr{D}$, since mining on the entire dataset is unfeasible. The parameters are: *images_per_classes* is the maximum number of images loaded for each randomly sampled class (of course we need at least two samples per class in order to compose a valid triplet); *mining_batch_size* the maximum value of $|\mathscr{B}|$, *landmarks* are the dataset classes; t, T and δ are thresholds and their purpose will be explained later. After the extraction of all the descriptors from the mining batch, the outer loop iterates along each feature vector and compute its nearest neighbours, then it starts generating the triplet with the corresponding image as query.

The inner loop iterates over the query ranking list. The hardest negative in the mining batch size is found (line **17**), then if it is not the nearest sample in the batch (line **18**), a random selection with the same probability is used to select the positive sample (line **19**, case **A**) as the element before in the ranking list. Note that in this case, $d(\mathbf{q}, \mathbf{p}) < d(\mathbf{q}, \mathbf{n})$, so the loss of this triplet is lesser than m. Alternatively (case **B**) the algorithm keeps with the search of the positive sample; however, if it does not find it within t-indices from the negative index n, the inner loop terminates (lines **9-10**). This early exit is motivated both to avoid searching a hard-positive sample and to control the complexity of the algorithm, avoiding useless iterations over the ranking list. After that the algorithm composes the triplet, its loss is computed and if its value is in the interval $(m - \delta, m + \delta)$, the triplet is appended to the triplet list, otherwise it is discarded (line **31, 33**). This filtering, paired with the random selection in line **18** makes sure that each triplet is balanced, without easy or hard samples, and with the average loss of the triplet list approximately equal to m. At the end of the outer loop, we have a list of up to *mining_batch_size* triplets that is further filtered with two thresholds. For the first one, only one triplet per class is maintained (with respect to the query/positive) to reduce redundant information (e.g. two different triplets where positive and anchor are swapped and with the same negative) and reduce overfitting risk of learning similar triplets. The second

thresholding (line **39**) was fundamental in our experiments, since this practice forces the training procedure to mine the triplets more often, avoiding the problem of **triplets aging**. The issue of triplet aging is when, after some weight updates via backpropagation, a triplet has a lower loss than before, and thus has little or no information worth to be learned. In our experiments, the *mining_batch_size* used in Algorithm 1 is 8000, *images_per_classes* is 15, t is 20, $\delta = 0.04$ and T was 240. The triplets are returned as minibatches of 6, while the mining was performed every 40 iterations (weight updates).

It can be shown that, ignoring the features extraction and Nearest-Neighbours steps, assuming $t' = images\_per\_classes$ and $n = |\mathscr{B}| = mining\_batch\_size$ the complexity of Algorithm 1 is $O(n)$. In fact, the inner loop has first to find the hard-negative, which in the worst case is found at the iteration $t' + 1$; with probability 0.5 the positive is selected with no additional costs, otherwise the loop continues the search, but if it is not found after t iterations, the algorithm exits from the inner loop, for a cost of $O(t + t')$. The loss computation (line **26**) could be optimized by reusing the k-NN metrics. Since the inner loop is repeated n times, the overall complexity is $O((t + t')n)$. In our setup, $t' < t$ and $t << n$, so the cost can be reduced to $O(n)$.

Training of the network is performed using Google Landmark V2 dataset.[3] In particular we use the train split of the "cleaned" version[4] presented in [33], that contains 1,580,470 images and 81,313 labels. In our experiments one epoch was equivalent to 400 steps, so the mining process was performed every 40 iterations, ten times for epoch, to account for the fact that descriptors may change greatly and the triplet aging we mentioned, especially during the initial training. The network has been trained using the Adam [23] optimizer; we performed the LR warm-up, starting from 10^{-6} increasing linearly until 10^{-5} at the 2000th iteration; then, the LR was decreased of one order of magnitude at the epoch 120; at the epoch 240, we stopped the training. We set the Batch Normalization layers of ResNet architectures to inference mode even during training, to reduce overfitting. In our experiments, the training images have been resized to resolution 336×336 pixels, regardless to the original aspect-ratio, but the final architecture has been trained at resolution 512×512, proving that an high training resolution is highly beneficial to the final result, as observed in [7,43], since at training time the weights are optimized for seeing patterns at closer scale and resolution to testing (Sect. 4.4.3). The PCA-Whitening has been trained on a subset of 65K descriptors extracted from the training images.

[3] https://github.com/cvdfoundation/google-landmark.

[4] https://www.kaggle.com/confirm/cleaned-subsets-of-google-landmarks-v2.

4 Experiments

For the convolutional part of the network we thoroughly evaluate two popular architectures, commonly used in other competing approaches, i.e. VGG16 and ResNet, but other architectures can be plugged, such as the recent ResNeSt, etc.

4.1 Datasets and metrics

The majority of the experiments have been made using Revisited Oxford5K and Revisited Paris6k datasets [37], using the *Medium* and *Hard* setups, and using ResNet backend, as this combination has become the most commonly used in the most recent state-of-the-art methods. To further extend comparison with other methods we have compared our method also using Oxford5k dataset [36] and Paris6k dataset [35], using VGG16 as backend. It has to be noted that the Revisited version of the datasets is more challenging. We use mean Average Precision (mAP, higher values are better) to evaluate the performance, as this is the metric commonly used to compare different approaches on the selected datasets. mAP is obtained computing the mean of the average precision (AP) of the set of queries used in the various datasets used in the following experiments. AP is the area under the precision-recall curve obtained plotting Precision (P) as a function of Recall (R), and accounts for the ranking of the retrieved results, so that a system that retrieves correct elements in the first positions of the result list is scores better. In particular, we use the following formulation:

$$AP = \frac{\sum_k P(k) \times rel(k)}{\#rel\_images}$$

where k is the rank in the sequence of retrieved documents, $P(k)$ is the precision at cut-off k in the list, $rel(k)$ is an indicator function equaling 1 if the image at rank k is a relevant image, and $\#rel\_images$ is the number of relevant images for the query.

4.2 Test configuration

In our experiments we used a multi-resolution approach, using an image pyramid to produce a representation at multiple scales. Considering the base resolution of 512×512, our final descriptor is produced by averaging the whitened descriptors at resolutions $\{\frac{2}{3}, 1, \frac{3}{2}\}$ of the chosen base resolution, regardless of the aspect-ratio, and then applying a final L2-normalization. However, we added a simple preprocessing step for the query images: since some test queries would have been heavily distorted by the rescaling operation, we first added a white padding to produce a squared image (thus the actual content is not distorted), and then we resized

Table 1 Effects of multi-scale pooling (mAP)

Pooling	ROxf (M)	ROxf (H)	RPar(M)	RPar(H)
3×3	61.1	36.4	78.9	59.5
2×2	60.8	39.5	78.2	59.0
Both	61.3	39.5	79.0	59.9

Table 2 Effects of using multi-scale images (mAP)

Scaling factors	ROxf (M)	ROxf (H)	RPar(M)	RPar(H)
$\{1\}$	63.0	36.7	80.9	62.4
$\{\frac{2}{3}, 1\}$	64.3	38.2	82.2	64.1
$\{1, \frac{3}{2}\}$	62.9	38.1	80.5	61.9
$\{\frac{2}{3}, 1, \frac{3}{2}\}$	64.6	39.8	82.5	64.7

Best results for each dataset are highlighted in bold

the queries using the base resolution multiplied by a scaling factor equal to 0.7; the motivation for this is to cope with the scale differences of the query images and the database images.

4.3 Multi-scale pooling and image resolutions

In this first set of experiments we use ResNet101 as FCNN. The experiments reported in Table 1 evaluate the effects of the first contribution of this work, i.e. using multi-scale features obtained from two max-pooling layers before the NetVLAD layer. Results show that using both 2×2 and 3×3 pooling improve the performance. A single resolution image is used as input, resized at resolution 336×336, without applying the PCA-Whitening. It must be noted that all the results improve upon the standard NetVLAD pooling [1] as shown in Table 7, showing the benefit of the two-step local CNN features aggregation.

Different resolutions may provide different clues regarding the appearance of objects in the scene. Hence, we extract and combine features at different resolutions, improving the performance of the multi-scale pooling. In the experiments reported in Table 2 we evaluate using different image resolutions at test time, evaluating the best combination on multiple datasets. Images are resized to, starting from a base resolution of 512×512, to different combination of the image pyramid, produced multiplying the base resolution to the scaling factors in $\{\frac{2}{3}, 1, \frac{3}{2}\}$; the images are resized regardless of aspect ratio, also enabling feature extraction in batches. The multi-resolution column reports the combination of scaling factor used. Results show that image multi-resolution improves the performance of the retrieval.

4.4 Ablation studies, dimensionality reduction and network architectures

In this set of experiments we use the more modern ResNet and the very recent ResNeSt networks.

4.4.1 Ablation studies

In these tests we evaluate the improvements of our modifications over the original NetVLAD configuration. All the models from Table 3 have been trained at resolution 336×336. Each model is tested after averaging the three multi-resolution embeddings, then applying PCA-Whitening to reduce the dimensionality of the resulting descriptor to 2048. In our setup the split of the max-pooled descriptors is equal to a 4-Split, i.e. we split the ResNet final descriptor into four descriptors of dimensionality 512. We show how applying the ReLU activation is fundamental for obtaining a representative descriptor (1st row vs. 2nd row) especially to deal with the more challenging ROxford5k dataset, despite what was reported in [1] where it was believed that ReLU activations are not meaningful in image retrieval, since the networks were trained for ImageNet classification. In the 3rd row, is shown how the addition of the two scale max-poolings is simple yet effective. Proved the effectiveness of the poolings and the convolutional activations, in the last rows the results without splitting the final tensor are presented. For these tests, we reduced the NetVLAD vocabulary to avoid the

Table 3 Effectiveness of the architectural improvements (mAP, using ResNet101 trained at 336×336 resolution)

ROxf (M)	ROxf (H)	RPar(M)	RPar (H)	ReLU	N-Split	Poolings	Vocabulary dim.	Feat. dim.	VLAD dim.
66.2	42.8	**80.6**	**61.4**	•	•	•	64	512	32K
62.8	37.9	79.2	60.3		•	•	64	512	32K
65.8	41.7	78.6	59.1	•	•		64	512	32K
65.9	**43.1**	79.1	59.7	•		•	32	2048	65K
58.9	30.6	76.5	56.5	•		•	64	512*	32K
66.2	40.9	78.1	58.4	•		•	16	2048	32K
62.9	38.5	78.9	60.2			•	32	2048	65K
61.3	33.8	79.0	60.5				32	2048	65K

Table 4 Scores varying the PCA final dimension (mAP)

PCA dim.	ROxf (M)	ROxf (H)	RPar(M)	RPar (H)
4096	66.6	43.7	80.7	61.3
2048	66.2	42.8	80.6	61.4
1024	66.0	41.0	79.6	60.0
512	63.0	35.4	79.2	59.0
256	61.6	33.6	77.5	56.7

dimensional explosion of the VLAD descriptor. We reduced the vocabulary first to 32 words, then to 16, thus yielding a descriptor of size comparable with our proposed setup. Comparison with the first three rows of the table clearly show the benefit of the splitting. In row 5, where the feature dimension is marked with * and is equal to 512, we did not extracted the features from the last layer but from the last bottleneck layer with dimensionality 512 to avoid the splitting: these results prove how is better achieving dimensionality reduction with the split operation rather than using an intermediate layer. In the penultimate row, is employed the configuration with just the double-scale max-poolings, as in [51]. A baseline method, that simply uses NetVLAD without any of the proposed improvements is reported in the last line of the table.

4.4.2 Triplet mining

We tested our proposed triplet mining procedure versus several baseline sampling methods on the ROxf and RPar datasets with the same ResNet architecture. In particular, we compare two different image sampling for positive images.

Descriptor distance We chose as positive the image that has the lowest descriptor distance relative to the query. That is $\mathbf{p} = argmin_{p \in \mathscr{D}}\ d(\mathbf{q}, \mathbf{p})$ where basically we take the nearest positive image in the feature space.

Maximum inliers Similarly as in [39], and as typically used in BoW methods, we exploit 3D information to choose the positive image, independently of the CNN descriptor. The image that has the highest intersection of 3D points with the query is chosen. Formally, $\mathbf{p} = argmax_{p \in \mathscr{D}}\ |\mathscr{P}(q) \cap \mathscr{P}(p)|$

Results Results are reported in Table 5. We can observe that our triplet mining strategy obtains the best performance compared to the baselines. In a qualitative inspection, our method consistently proposes more challenging positive examples with increased variability of viewpoints.

4.4.3 PCA-whitening dimensionality reduction

NetVLAD descriptors have a high dimensionality, thus it is required to reduce it to obtain a more manageable descriptor. In Table 4 we evaluate the effect of PCA-based compression on the final descriptor, varying from 4096 to 256 dimensionality. It can be observed how keeping more eigenvectors from the PCA increases the descriptor performance; under 1024 dimensions there is a major information loss, noticeable especially when testing with the ROxford5k dataset. However, for our tests we kept the 2048-dim compression for the sake of the comparison with other state-of-the-art descriptors.

Table 5 Performance of different sampling methods (mAP)

Method	ROxf (M)	ROxf (H)	RPar(M)	RPar(H)
Descr. distance	63.37	39.82	80.44	62.35
Max. inliers	67.68	44.51	80.18	61.92
Our mining	73.18	53.03	82.59	64.38

4.4.4 Trainings at higher resolution with modern architectures

We previously stated how training at high resolution is proven to be fundamental [7,43] for Image Retrieval. It is common knowledge and practice in Image Retrieval, that extracting features from images at higher resolution produces more meaningful descriptors. Still, this is sub-optimal: it can be explained by the fact that, if the weights are fine-tuned with images at lower resolution respect to the testing phase, the model does not have the chance to learn in-depth some patterns, which disappear with the downsampling, instead. Moreover, it should be considered that the receptive field would be larger during training, thus the network learns patterns at larger scale with respect to the testing phase; however, this is partially addressed with the multi-resolution descriptor. On these premises, and after selecting our more prominent setup (4-Split + ReLU + Max-Poolings), we proceed for a more time-expensive experiment training the networks at resolution 512×512. Finally, we test a very recent ResNet variant, the ResNeSt101 and ResNeSt50 [58], that have shown improved results in different tasks, from segmentation to classification; our experiments show that this architecture heavily outperforms its more traditional version also for the image retrieval task. In our experimental setup, the ResNeSt101 and ResNeSt50, respectively, completed the features extraction from the test datasets in about 120% and 85% of the total time employed by the original ResNet101. These results are shown in Table 6.

4.5 Comparison with state-of-the-art

In these experiments we evaluate the proposed method with current state-of-the art methods on all the datasets and using the three CNN architectures used in the two previous sets of experiments.

Table 6 Scores varying network architecture and training resolution (mAP)

Net	Train res.	ROxf (M)	ROxf (H)	RPar(M)	RPar (H)
ResNeSt101	512	75.3	55.4	86.9	72.5
ResNeSt101	336	75.2	53.4	84.8	68.1
ResNeSt50	512	72.0	51.6	81.7	64.9
ResNeSt50	336	71.3	50.3	82.3	64.7
ResNet101	512	70.9	47.3	82.1	64.7
ResNet101	336	66.2	42.8	80.6	61.4

Table 7 Comparison with state-of-the-art methods (mAP) using VGG16

Method	Oxford5k	Paris6k
Our method	86.0	88.7
Our method (no crop)	<u>87.3</u>	**91.5**
MS-NetVLAD (no crop) [51]	83.8	<u>89.3</u>
GeM [39]	**87.9**	87.7
SAB [25]	74.3	86.9
SRCS [45]	74.6	85.9
R-MAC [14]	83.1	87.1
NetVLAD [1]	71.6	79.7
Fisher-Vector [32]	81.5	82.4
BoW-CNN [29]	73.9	82.0
R-MAC [49]	66.9	83.0
CroW [22]	70.8	79.7
PWA [55]	72.0	82.3
LS [29]	74.2	82.0
SpoC [4]	68.1	78.2

Best results are highlighted in bold and second best results are underlined

Table 7 reports a comparison of methods that use VGG networks tested on Oxford5k and Paris6k; this combination of architecture and dataset has been superseded in the most recent works and we report it because it allows to provide a more extensive comparison. Results of our method have been obtained using multi-resolution (see Sect. 4.2) and the VGG network has been trained at resolution 336×336 this time using the LR warm-up and keeping the activation functions after the final convolutional layer. The proposed method obtains the best result when performing queries without using image crops, as done in [51]; in this case it has state-of-the-art performance on Paris6k and second best result on Oxford5k. Instead, when using query crops as in the other competing approaches, it consistently obtains the second best result in both datasets, unlike the other two best methods that obtained the best result only in one of the two datasets and the third best result in the other.

Table 8 reports a comparison of methods using the more challenging Revisited Oxford5k and Revisited Paris6k datasets, using the Medium and Hard setups. These combinations of datasets and setups are used in the more recent works. Our results have been obtained using multi-resolution (see Sect. 4.2) and all the three proposed techniques (ReLU, N-Split and multi-resolution pooling) analyzed in Table 3. The competing methods use ResNet networks, so they should be compared with our results reported on line 2. Our proposed method outperforms the most recent approach of [31] in 3 of the 4 setups, and obtains the second best result in ROxford5k Hard.

Table 8 Comparison with the state of the art (mAP) using ResNet. Best results are highlighted in bold and second best results are underlined. SpoC values reported from [18]

Method	ROxf (M)	ROxf (H)	RPar(M)	RPar (H)
Ours (ResNeSt101)	**75.3**	**55.4**	**86.9**	**72.5**
Ours (ResNet101)	<u>73.2</u>	<u>53.0</u>	<u>82.6</u>	64.4
SOLAR [31]	69.9	47.9	81.6	<u>64.5</u>
DELG [7]	69.7	45.1	81.6	63.4
BE [26]	67.5	41.8	80.3	61.2
OLMANS [61]	65.7	40.4	76.9	55.4
IRT_R [10]	55.1	28.3	72.7	49.6
GeM+CA [18]	67.3	42.6	77.5	56.5
AGeM [15]	67.0	40.7	78.1	57.3
HesAff-HardNet [28]	65.6	76.9	43.1	55.4
ASDA [56]	66.4	38.5	71.6	47.9
AP-Loss [43]	67.5	42.8	80.1	60.5
GeM [39]	64.7	38.5	77.2	56.3
SuperLoss [8]	62.7	38.1	77.0	56.5
R-MAC [14]	60.9	32.4	78.9	59.4
NetVLAD [1]	37.1	13.8	59.8	35.0
SpoC [4]	39.8	12.4	69.2	44.7

It is worth noting the very large performance boost provided by ResNeSt architecture shown in line 1, that suggests that this novel architecture should be used instead of ResNet for image retrieval tasks. Using ResNeSt50 would still provide results that are better than ResNet101 (see Table 6) but with smaller computational costs.

5 Conclusions

In this paper we presented a novel multi-scale local CNN features pooling that, by exploiting end-to-end learning on a Siamese network, is able to learn an effective images representation. A feature splitting step allows to avoid dimensionality explosion due to VLAD coding. The approach can be used with different network architecture blocks for feature extraction as shown in the experiments performed using VGG16, ResNet and the recent ResNeSt. The proposed

approach has been thoroughly analysed and compared with a large number of competing approaches and obtains state-of-the-art results in standard and challenging datasets for image retrieval. This result is obtained also thanks to a novel triplet mining procedure that is able to diversify triplets based on their difficulty and focus the learning on the most significative ones.

Despite using one of the most recent backbones and an improved triplet mining algorithm, the task at hand still suffer from the problem of semantic gap which is evident from the experimental results. Our future work will then consider the applicability of the proposed method to visual transformer backbones and the extension to include temporal aspects to address near duplicate video retrieval and video retrieval using query frames.

Acknowledgements We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. This work was supported by European Union's Horizon 2020 research and innovation programme under grant number 951911 - AI4Media

References

1. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: CNN architecture for weakly supervised place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR, NetVLAD (2016)
2. Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: From generic to specific deep representations for visual recognition. In: Proceedings of CVPR Workshops (2015)
3. Babenko, A., Lempitsky, V.: Aggregating deep convolutional features for image retrieval. In: Proceedings of ICCV (2015)
4. Babenko, A., Lempitsky, V.: Aggregating local deep features for image retrieval. In: Proceedings of ICCV, pp. 1269–1277 (2015)
5. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Proceedings of ECCV (2014)
6. Ballan, L., Bertini, M., Uricchio, T., Del Bimbo, A.: Social media annotation. In: Proceedings of CBMI, pp. 229–235 (2013)
7. Cao, B., Araujo, A., Sim, J.: Unifying deep local and global features for image search. In: Proceedings of ECCV, Springer, pp. 726–743 (2020)
8. Castells, T., Weinzaepfel, P., Revaud, J.: Superloss: a generic loss for robust curriculum learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 4308–4319. Curran Associates Inc, New York (2020)
9. Delhumeau, J., Gosselin, P.-H., Jegou, H., Pérez, P.: Revisiting the VLAD image representation. In: Proceedings of ACM MM (2013)
10. El-Nouby, A., Neverova, N., Laptev, I., Jégou, H.: Training vision transformers for image retrieval. arXiv preprint arXiv:2102.05644 (2021)
11. Ercoli, S., Bertini, M., Del Bimbo, A.: Compact hash codes for efficient visual descriptors retrieval in large scale databases. IEEE Trans. Multimedia (TMM) **19**(11), 2521–2532 (2017)
12. Gong, Y., Wang, L., Guo, R., Lazebnik, S.: Multi-scale orderless pooling of deep convolutional activation features. In: Proceedings of ECCV (2014)
13. Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Learning global representations for image search. In: Proceedings of ECCV, Deep image retrieval (2016)
14. Gordo, A., Almazan, J., Revaud, J., Larlus, D.: End-to-end learning of deep visual representations for image retrieval. Int. J. Comput. Vis. **124**(2), 237–254 (2017)
15. Gu, Y., Li, C., Xie, J.: Attention-aware generalized mean pooling for image retrieval. arXiv preprint arXiv:1811.00202 (2018)
16. Hausler, S., Garg, S., Xu, M., Milford, M., Fischer, T.: Patch-NetVLAD: multi-scale fusion of locally-global descriptors for place recognition. In: Proceedings of CVPR, pp. 14141–14152 (2021)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of CVPR, pp. 770–778 (2016)
18. Hu, Z., Bors, A.G: Conditional attention for content-based image retrieval. In: Proceedings of BMVC, York (2020)
19. Iscen, A., Tolias, G., Gosselin, P.-H., Jégou, H.: A comparison of dense region detectors for image search and fine-grained classification. IEEE Trans. Image Process. **24**(8), 2369–2381 (2015)
20. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE Trans. Pattern Anal. Mach. Intell. **34**(9), 1704–1716 (2012)
21. Jégou, H., Douze, M., Schmid, C.: Improving bag-of-features for large scale image search. Int. J. Comput. Vis. **87**(3), 316–336 (2010)
22. Kalantidis, Y., Mellina, C., Osindero, S.: Cross-dimensional weighting for aggregated deep convolutional features. In: Hua, G., Jégou, H. (eds.) Proceedings of ECCV Workshops, Springer, Cham, pp 685–701 (2016)
23. Kingma, D.P., Ba, J.: A method for stochastic optimization. In: Proceedings of ICLR, Adam (2014)
24. Li, X., Uricchio, T., Ballan, L., Bertini, M., Snoek, C.G.M., Bimbo, A.D.: Socializing the semantic gap: a comparative survey on image tag assignment, refinement, and retrieval. ACM Comput. Surv. **49**(1), 1–39 (2016)
25. Li, X., Jin, K., Long, R.: End-to-end semantic-aware object retrieval based on region-wise attention. Neurocomputing **359**, 219–226 (2019)
26. Martínez-Cortés, T., González-Díaz, I., Díaz de María, F.: Training deep retrieval models with noisy datasets: bag exponential loss. Pattern Recognit. **112**, 107811 (2021)
27. Mikulik, A., Perdoch, M., Chum, O., Matas, J.: Learning vocabularies over a fine quantization. Int. J. Comput. Vis. **103**(1), 163–175 (2013)
28. Mishkin, D., Radenović, F., Matas, J.: Learning affine regions via discriminability. In: Proceedings of ECCV, Repeatability is Not Enough (2018)
29. Mohedano, E., McGuinness, K., O'Connor, N.E., Salvador, A., Marques, F., Giro-i Nieto, X.: Bags of local convolutional features for scalable instance search. In: Proceedings of ACM ICMR (2016)
30. Morère, O., Lin, J., Veillard, A., Duan, L.-Y., Chandrasekhar, V., Poggio, T.: Nested invariance pooling and RBM hashing for image instance retrieval. In: Proceedings of ACM ICMR (2017)
31. Ng, T., Balntas, V., Tian, Y., Mikolajczyk, K.: Second-order loss and attention for image retrieval, SOLAR (2020)
32. Ong, E.-J., Husain, S., Bober, M.: Siamese network of deep Fisher-vector descriptors for image retrieval. arXiv preprint arXiv:1702.00338 (2017)
33. Ozaki, K., Yokoo, S.: Large-scale landmark retrieval/recognition under a noisy and diverse dataset. arXiv preprint arXiv:1906.04087 (2019)
34. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher kernel for large-scale image classification. In: Proceedings of ECCV, (2010)
35. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: improving particular object retrieval in large scale image databases. In: Proceedings of CVPR (2008)

36. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: Proceedings of CVPR (2007)
37. Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Large-scale image retrieval benchmarking. In: Proceedings of CVPR, Revisiting Oxford and Paris (2018)
38. Radenović, F., Tolias, G., Chum, O.: Unsupervised fine-tuning with hard examples. In: Proceedings of ECCV, CNN Image Retrieval Learns from BoW (2016)
39. Radenović, F., Tolias, G., Chum, O.: Fine-tuning CNN image retrieval with no human annotation. IEEE Trans. Pattern Anal. Mach. Intell. **41**(7), 1655–1668 (2018)
40. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for visual recognition. In: Proceedings of CVPR Workshop of DeepVision (2014)
41. Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: A baseline for visual instance retrieval with deep convolutional networks. ITE Trans. Media Technol. Appl. **4**, 12 (2014)
42. Mopuri, K.R., Babu, R.V.: Object level deep feature pooling for compact image representation. In: Proceedings of CVPR Workshops (2015)
43. Revaud, J., Almazan, J., Rezende, R.S., de Souza, C.R.: Training image retrieval with a listwise loss. In: Proceedings of ICCV, Learning with Average Precision (2019)
44. Schuster, R., Wasenmuller, O., Unger, C., Stricker, D.: SDC-Stacked dilated convolution: a unified descriptor network for dense matching tasks. In: Proceedings of CVPR, pp. 2556–2565 (2019)
45. Shi, X., Qian, X.: Exploring spatial and channel contribution for object based image retrieval. Knowl. Based Syst. **186**, 104955 (2019)
46. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
47. Sivic, J., Zisserman, A.: Video google: a text retrieval approach to object matching in videos. In: Proceedings of ICCV (2003)
48. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell. **22**(12), 1349–1380 (2000)
49. Tolias, G., Sicre, R., Jégou, H.: Particular object retrieval with integral max-pooling of CNN activations. In: Proceedings of ICLR (2016)
50. Uricchio, T., Bertini, M., Seidenari, L., Del Bimbo, A.: Fisher encoded convolutional Bag-of-Windows for efficient image retrieval and social image tagging. In: Proceedings of ICCV International Workshop on Web-Scale Vision and Social Media (VSM) (2015)
51. Vaccaro, F., Bertini, M., Uricchio, T., Del Bimbo, A.: Image retrieval using multi-scale CNN features pooling. In: Proceedings of ACM ICMR (2020)
52. Wang, X., Zhu, L., Yang, Y.: T2VLAD: global-local sequence alignment for text-video retrieval. In: Proceedings of CVPR, pp. 5079–5088 (2021)
53. Wu, C.-Y., Manmatha, R., Smola, A.J., Krähenbühl, P.: Sampling matters in deep embedding learning. In: Proceedings of ICCV (2017)
54. Xie, L., Hong, R., Zhang, B., Tian, Q.: Image classification and retrieval are ONE. In: Proceedings of ACM ICMR (2015)
55. Xu, J., Shi, C., Qi, C., Wang, C., Xiao, B.: Unsupervised part-based weighting aggregation of deep convolutional features for image retrieval. In: Proceedings of AAAI (2018)
56. Xu, J., Wang, C., Shi, C., Xiao, B.: Weakly supervised soft-detection-based aggregation method for image retrieval. In: CoRR. arXiv preprint arXiv:1811.07619 (2018)
57. Yue-Hei Ng, J., Yang, F., Davis, LS.: Exploiting local features from deep networks for image retrieval. In: Proceedings of CVPR Workshops (2015)
58. Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Mueller, J., Manmatha, R.., Li, M., Smola, A: ResNeSt: split-attention networks (2020)
59. Zheng, L., Yang, Y., Tian, Q.: SIFT meets CNN: a decade survey of instance retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **40**(5), 1224–1244 (2017)
60. Zheng, L., Zhao, Y., Wang, S., Wang, J., Tian, Q.: Good practice in CNN feature transfer. arXiv preprint arXiv:1604.00133 (2016)
61. Zhou, J., Ying, W.: Learning visual instance retrieval from failure: Efficient online local metric adaptation from negative samples. IEEE Trans. Pattern Anal. Mach. Intell. **42**(11), 2858–2873 (2019)

Federico Vaccaro received the Bachelor Degree at the University of Florence in Computer Engineering in 2018, with his thesis on context-based image retrieval techniques based on deep learning. He worked for one year as a research scholar at the Media Integration and Communication Center (MICC) laboratory of the University of Florence, focusing on Image Retrieval systems and state-of-the-art techniques, for which he received the best poster award at ACM ICMR 2020. In 2021, he obtained the Master Degree in Computer Engineering at the University of Florence with a thesis on deep learning-based real-time image quality improvement using GANs and UNets. Currently, he is working as consultant for Data Reply IT, mostly on thematics related to quantum and accelerated computing.

Marco Bertini received the Laurea degree in electronic engineering from the University of Florence, Florence, Italy, in 1999, and the Ph.D. degree in 2004. He is the Director of the Media Integration and Communication Center, University of Florence, and is an Associate Professor with the School of Engineering, University of Florence. He is author of 28 journal papers and more than 120 peer-reviewed conference papers, with an h-index: 33 (according to Google Scholar). His interests focus on multimedia and social media analysis, video quality enhancement and CBIR. On these subjects, he has addressed semantic analysis, content indexing and annotation, semantic retrieval, and smart video coding and enhancement.

Tiberio Uricchio received his PhD in Computer Engineering from the University of Florence in 2016, with his thesis "Image understanding by socializing the semantic gap", awarded as the best in the technological class by the University of Florence and published as a book in 2018. He is author of more than 30 publications in international scientific journals and conferences on computer vision and multimedia. His research interests include understanding images and videos, multimedia, deep learning and video surveillance. Since 2020, he is CEO and co-founder of the startup Small Pixels, on a mission to improve perceptual quality of videos.

Alberto Del Bimbo is Full Professor at the University of Firenze, Italy, and the Director of MICC Media Integration and Communication Center. He is the author of over 350 scientific publications in computer vision and multimedia and principal investigator of technology transfer projects with industry and governments. He was the Program Chair of ICPR 2012, ICPR 2016, and ACM Multimedia 2008, and the General Chair of IEEE ICMCS 1999, ACM Multimedia 2010, ICMR 2011 and ECCV 2012. He is the General Chair of the forthcoming ICPR 2020. He is the Editor in Chief of ACM TOMM Transactions on Multimedia Computing Communications and Applications and Associate Editor of Multimedia Tools and Applications and Pattern Analysis and Applications journals. He was Associate Editor of IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Multimedia and Pattern Recognition and also served as the Guest Editor of many Special Issues in highly ranked journals. Prof. Del Bimbo is IAPR Fellow and ACM Distinguished Scientist and is the recipient of the 2016 ACM SIGMM Award for Outstanding Technical Contributions to Multimedia Computing Communications and Applications.