

Deep Aggregation of Regional Convolutional Activations for Content Based Image Retrieval

Konstantin Schall, Kai Uwe Barthel, Nico Hezel, and Klaus Jung

Visual Computing Group, HTW Berlin

Berlin, Germany

{schallk, barthel, hezel, jungk}@htw-berlin.de

Abstract—One of the key challenges of deep learning based image retrieval remains in aggregating convolutional activations into one highly representative feature vector. Ideally, this descriptor should encode semantic, spatial and low level information. Even though off-the-shelf pre-trained neural networks can already produce good representations in combination with aggregation methods, appropriate fine tuning for the task of image retrieval has shown to significantly boost retrieval performance. In this paper, we present a simple yet effective supervised aggregation method built on top of existing regional pooling approaches. In addition to the maximum activation of a given region, we calculate regional average activations of extracted feature maps. Subsequently, weights for each of the pooled feature vectors are learned to perform a weighted aggregation to a single feature vector. Furthermore, we apply our newly proposed NRA loss function for deep metric learning to fine tune the backbone neural network and to learn the aggregation weights. Our method achieves state-of-the-art results for the INRIA Holidays data set and competitive results for the Oxford Buildings and Paris data sets while reducing the training time significantly.

Index Terms—Computational and artificial intelligence, Multi-layer neural network, Image retrieval, Content-based retrieval, Machine learning, Feature extraction, Machine learning algorithms, Nearest neighbor searches, Computer vision

I. INTRODUCTION

With the rising amount of visual data produced and shared on the internet, the task of grouping and searching for related content becomes more challenging and more important. Information about millions of images has to be stored in very compact representations to be able to efficiently accomplish such tasks. Recent works have shown that representations extracted by a deep neural network can be used to form excellent feature vectors, which encode visual similarity of images in a high dimensional space [4], [6], [27]. A convolutional neural network (CNN) often maps visually similar images to similar feature vectors whereas the extracted feature vectors are different for dissimilar images. Since these feature vectors are represented in floating point numbers, they can be easily compared by simple distance metrics such as L^2 , which makes them suited for a wide range of instance level image retrieval tasks. Most neural networks used for image retrieval however, have been trained to classify images into a fixed number of categories. Even though these networks already show good retrieval results [22], performance can be improved by applying a model that has been specifically trained to rank images by their similarity as shown in [6] and [18].

Deep metric learning forms a sub-field of deep learning, where the goal is to find a representation space, in which visual resemblance of images is perfectly encoded by spatial relationships in the feature vector space. In these approaches, optimization occurs directly on the extracted embedding vectors. Distances between samples of the same class are required to be smaller than distances between entities of different classes [21], [23]–[25]. Metric learning methods therefore use loss functions, which are trying to teach a neural network to generate such a representation space. Networks trained in this way are better suited for the task of image retrieval [25]. The Triplet loss function [21] is the most prominent example. In [6] the authors achieve excellent retrieval results by building an end-to-end trainable system which extracts regional maximum activations from convolutional feature maps similar to R-MAC [27]. They follow the ideas described in [27] and aggregate these activations to one high dimensional feature vector. The network is trained with the Triplet loss function [21] using the landmarks data set from [4] in one of several training steps. The authors scale images to 800 pixels at the larger dimension during training, which makes the whole process very computationally expensive due to GPU memory limitations.

In [20] we proposed the concept of *Nonlinear Rank Approximation* (NRA) loss. This loss function achieves very good results and outperforms the Triplet loss on a wide range of retrieval tasks significantly. In this paper we propose to apply the NRA loss function to train an end-to-end retrieval network similar to [6]. Since the NRA loss works best if batches are built from several classes containing multiple samples, large images cannot be used during training. However, we still can use relatively large image sizes for feature extraction due to the fully convolutional nature of the network and achieve better retrieval results with smaller resolutions than used in comparable work. Furthermore, we argue that the sum aggregation of the chosen number of regions extracted with the R-MAC method [27] does not fully take the scale and size of different regions into account. Instead we apply a small aggregation network on top of the backbone CNN, which learns the importance of each of the regions and assigns it an individual weight. The total system is trainable in under 24 hours on a single NVIDIA GTX 1080 Ti GPU with 12 GB memory, where 64 images fit into one single batch using the ResNet50 V2 architecture [7] as backbone CNN.

II. RELATED WORK

In this section we first describe the principle of the R-MAC descriptor, followed by an explanation of similar end-to-end trainable deep learning based image retrieval methods. Finally we explain the importance of the loss function, which is used to train the retrieval network.

A. The R-MAC descriptor

In [27] Tolias et al. describe a way to incorporate spatial information extracted with a convolutional neural network into an image’s feature vector. Usually the last convolutional layer of the chosen backbone CNN is used to obtain a set of feature maps $\mathcal{X} = \{\mathcal{X}_i | i = 1 \dots \mathcal{C}\}$, where \mathcal{C} is the number of output channels in the said layer and the final feature vector will have \mathcal{C} dimensions. These feature maps have been aggregated in various ways prior to the R-MAC method as in [2], [3], [5] neglecting the spatial information available. In [27] a grid structure of variable scales is defined, consisting of 20 regions for three different scales $l = 1 \dots 3$. The region’s size for scale l is calculated by $2 \min(W, H)/(l + 1)$. The regions are uniformly chosen in a sliding window manner with the current scale. In the next step, the maximum activation value of each region and feature map is chosen to form 20 \mathcal{C} -dimensional feature vectors. These are furthermore L^2 -normalized, whitened with a PCA [10] and then L^2 -normalized again individually, before they are sum pooled to produce a single vector, the R-MAC descriptor.

B. End-to-End Networks

In [6] the Deep Image Retrieval (DIR) system is described which is based on the previously mentioned R-MAC approach [27]. The authors goal was to design a convolutional feature extractor pipeline as presented in [27] and train the resulting architecture in an end-to-end manner, since all applied operations are differentiable. Importantly, the authors claim that the Softmax Cross Entropy loss function is not optimal for the task of fine tuning for retrieval and apply the Triplet loss function [21] instead. However, since the used landmarks data set [4] seems to be noisy, i.e. images of the same category are visually different, they could not train the proposed network with the Triplet loss directly. Instead they first fine tune the used ResNet101 [7] in a traditional, classification approach and then perform a cleaning of the landmarks data set, based on a visual matching comparison of SIFT image descriptors [13]. This cleaning reduces the landmarks data set from 192 000 to 42 410 training images with 586 categories. The authors refer to this version of the data set as *Landmarks Clean*. Furthermore, as retrieval performance seems to increase for input images with larger sizes [6], [14], Gordo et al. decide to perform the last step of their training procedure, i.e. fine tuning with Triplet loss with the landmarks clean data set, with images of relatively large size. Specifically, each training image is resized to 800 pixels at the larger side. However, one single image of this size results in approximately 7.5 GB GPU memory, which limits training to one image per batch on most GPUs. To overcome this limitation the authors introduce a

modified sequential training procedure. With this modification, they are indeed able to train the proposed architecture and achieve outstanding retrieval results at the price of one week of total training time. At the end of the training, the authors further improve their models performance by extracting feature vectors from images in multiple resolutions, namely 550, 800 and 1050 pixels at the larger side. Those feature vectors are L^2 -normalized, sum aggregated and L^2 -normalized again.

Noh et al. present DELF (DEep Local Feature) in [15]. A pre-trained ResNet50 network is used as the backbone CNN and two additional convolutional layers are added to model an attention functionality. The specific task of this part is to analyze the extracted set of feature maps and to assign an importance weight to each area. Images of size $224 \times 224 \times 3$ are used, which leads to feature maps of shape $7 \times 7 \times 2048$. Each of the 49 2048-dimensional feature vectors corresponds to a part in the input image which is specified by the network’s receptive field [11]. The attention part of the DELF architecture analyzes each of the 49 vectors and learns to assign a higher weight to a single area, if an interesting object is seen. For training, the authors further sum-aggregate the weighted combination of the 49 embeddings to form a single 2048-dimensional feature vector. Training is done in two steps. First, the original pre-trained ResNet50 is fine tuned with the same landmarks set as in [6]. In the second step the attention part is trained independently, i.e the weights of the backbone CNN are not further optimized. The Softmax Cross Entropy loss function is used during both training procedures.

C. Loss Functions for Retrieval

For an image classification task the set of feature maps \mathcal{X} is reduced to a \mathcal{C} -dimensional vector by averaging each activation map \mathcal{X}_i . This vector is further fed through a fully connected and a Softmax layer, where both have outputs equal to the number of classes of the training set. The output of the Softmax function can be seen as a probability distribution of the available classes. The Cross Entropy loss then tries to minimize the divergence between the ground truth and the predicted Softmax output. To achieve a low Softmax Cross Entropy, it is sufficient to divide the average pooled CNN representation space into as many regions as there are classes in the training (see Fig. 1 (a)). However, such a constellation is not suited for nearest neighbor search.

Since image retrieval quality is evaluated by the ranking order of nearest neighbors to a query image, the representation space should form tighter clusters for optimal search results. One loss function trying to achieve this goal is the Triplet loss function [21], which builds a set of three samples $\mathcal{T} = \{\mathbf{x}^a, \mathbf{x}^+, \mathbf{x}^-\}$ consisting of an anchor \mathbf{x}^a , one positive example \mathbf{x}^+ of the same class as the anchor and one negative example \mathbf{x}^- of a different class.

The loss value for a single triplet is calculated by Eq. (1) where D^2 denotes the L^2 -distance between two feature vectors $f(\mathbf{x})$ with $D_{i,j} = \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2$.

$$J_{\mathcal{T}} = \max(0, D_{a,+}^2 - D_{a,-}^2 + \alpha) \quad (1)$$

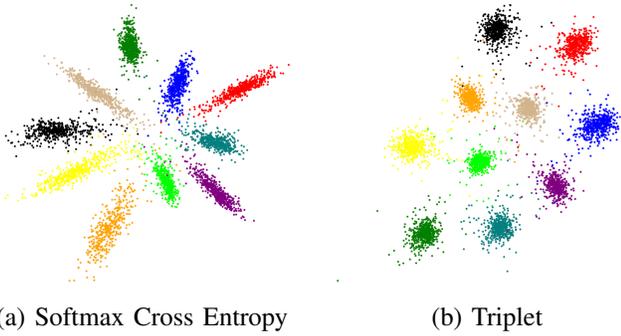


Fig. 1. Visualization of the 2D embedding space of MNIST numbers [12] after training the same network with two different loss functions. Representations produced by the Softmax Cross Entropy loss are not as suited for nearest neighbor search as the ones produced by the Triplet loss.

The Triplet loss tries to maximize the relative distance to a negative example and minimizes the distance to positive ones, ideally grouping images from the same class to clusters in the given representation space.

III. PROPOSED METHOD

A. Nonlinear Rank Approximation Loss

Following the assumption that a perfect clustering is achieved if and only if all distances to negative examples are larger than the maximum distance to positive examples, we introduced the Nonlinear Rank Approximation loss (NRA) in [20]. The main principles of this loss function will be briefly explained below.

NRA is taking the two most important distances w.r.t. an anchor into account: The maximum distance to the embedding of its positive examples of the same class and the minimum distance to its negative examples from a different class:

$$D_{i,\max}^+ = \max_j D_{i,j} \quad D_{i,\min}^- = \min_j D_{i,j} \quad (2)$$

To enable a search for those points, $m = k \cdot n$ samples x_i of the input data are used per batch, where k is the number of classes and n the number of samples per class. This step ensures each sample in the batch to have $n - 1$ positive and $m - n - 1$ negative examples. For each feature vector in the batch, the distances to all other feature vectors are determined. This yields a $m \times m$ distance matrix, where $D_{i,\max}$ is the maximum of its i^{th} row values and $D_{i,\min}$ is the minimum accordingly.

$$D_{i,\max} = \max_j D_{i,j} \quad D_{i,\min} = \min_j D_{i,j} \quad (3)$$

Instead of using this distance matrix directly, the approximated ranks $r_{i,j}$ are computed by min-max-scaling each row to the interval $[0, 1]$ individually:

$$r_{i,j} = \frac{D_{i,j} - D_{i,\min}}{D_{i,\max} - D_{i,\min}} \in [0, 1] \quad (4)$$

This scaling ensures a unified representation of the distances for each batch during training by approximating a ranking order, where the lowest possible rank is 0 and the highest

is 1. Subsequently, the approximated normalized ranks for the most distant embedding of the same class $r_{i,\max}^+$ and the closest embedding of a sample of a different class $r_{i,\min}^-$ are determined by

$$r_{i,\max}^+ = \frac{D_{i,\max}^+ - D_{i,\min}}{D_{i,\max} - D_{i,\min}} \quad r_{i,\min}^- = \frac{D_{i,\min}^- - D_{i,\min}}{D_{i,\max} - D_{i,\min}} \quad (5)$$

Next for each row these two ranks are transformed to similarities. Instead of calculating the similarity as $s = 1 - r$ a parametric nonlinear transfer function is used

$$w(\cdot; \alpha) : [0, 1] \rightarrow [0, 1] \quad (6)$$

$$w(r; \alpha) = \begin{cases} \frac{1}{2}(2r)^\alpha & r \in [0, \frac{1}{2}) \\ 1 - \frac{1}{2}(2(1-r))^\alpha & r \in [\frac{1}{2}, 1] \end{cases} \quad (7)$$

with $\alpha \in \mathbb{R}^+$. The parameter α controls the slope of $w(\cdot; \alpha)$ at $r = 0.5$. Thus, it determines the influence of rank errors. With increasing α falsely ranked negative images with a high similarity are stronger punished and correctly ranked positive images with a high similarity will result in a loss value closer to zero. An empirical study showed best results for $\alpha = 4$ [20]. Using $w(\cdot; \alpha)$ and omitting α in the following notation we define the similarity of a sample x_j to the anchor x_i by $s_{i,j} = 1 - w(r_{i,j})$. The final loss value for each batch is calculated as follows:

$$J = -\frac{1}{m} \sum_{i=1}^m \left(\log(s_{i,\max}^+ + \varepsilon) + \log(1 - s_{i,\min}^- + \varepsilon) \right) \quad (8)$$

Here $\varepsilon > 0$ is a small number that controls the maximum loss contribution if $s_{i,\max}^+ = 0$ or $s_{i,\min}^- = 1$. We use $\varepsilon = 10^{-4}$ throughout the following experiments.

B. Global and Regional Pooling

The R-MAC approach [27] aggregates maximum activations of different regions in convolutional feature maps into one final feature vector per image. However, the maximum activation of a region can be very high, even though the detected shape is only seen on a small area of the region. Furthermore, since many of the 20 regions have overlapping areas, the pooled max value will be the same in some of the areas. Calculating the average activation value of each region gives a more accurate representation of an image's convolutional activations but is more sensitive to changes in the input image's size. To better understand the differences of both pooling operations, we conduct a preliminary study and build different feature vectors from convolutional activation maps extracted with a ResNet50 v2 network [8] for different input image sizes. The network is pre-trained with images with 224 pixels on both sides from the ILSVRC2012 data set [19] and we use images from a rotated version of the INRIA Holidays data set [9] for evaluation.

The networks last convolutional layers are used as a feature map extractor and six different pooling techniques are compared in the preliminary study. The first is built by a global average pooling of the entire set of feature maps (Avg Global), whereas the second comes from the global max pooling operation (Max Global). The third represents the

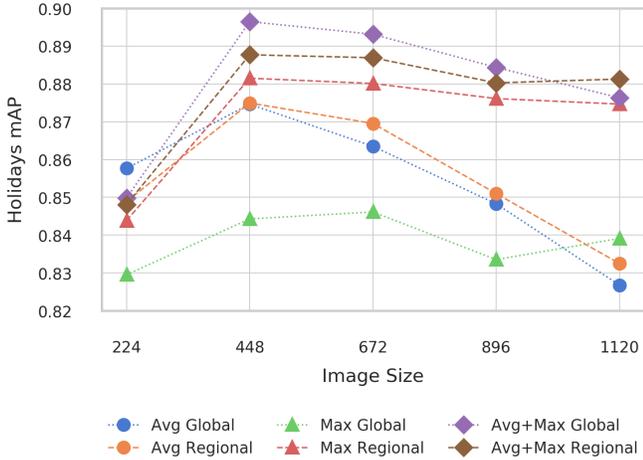


Fig. 2. Mean Average Precision scores for the Holidays data set and different aggregation techniques for varying input image sizes.

sum of those two globally pooled values (Avg+Max Global). Further, we build three regional pooled and aggregated feature vectors. One for the average pooled regional activations (Avg Regional), another for the max pooled regional activations (Max Regional) similar to R-MAC [27] and the last is created by a pooling of maximum and average values from each region (Avg+Max Regional). For the regional approaches we choose the 20 areas as used in [27]. In case of the combined regional approach, we obtain 20 max pooled and 20 average pooled values. Furthermore, we L^2 normalize each feature vector individually in the regional methods, sum pool over the regions and L^2 normalize again. With each of those six methods, we extract feature vectors for five image sizes $S \in \{224, 448, 672, 896, 1120\}$. Fig. 2 summarizes the results from this experiment.

Several important conclusions can be drawn from these results. Firstly, all aggregation methods benefit from an image size, twice as high as the images seen during training. Global max pooling performs poorly across all tests. The mean average precision scores drop drastically if using average pooling based methods for higher resolutions. The R-MAC method achieves good results, but is outperformed by the pooling techniques that use a combination of max and average pooling.

In this experiment, the global version of the combined max and average pooling method (Avg+Max Global) outperforms the regional approach (Avg+Max Regional) in most resolutions. We argue that this is due to the fact that the 40 pooled values are not equally important, but are handled this way during the sum aggregation. For example, the activations of individual regions should be assigned a weight according to their size and the average value seems to be much more important in the bigger regions compared to the maximum value. Furthermore, the global activation map should be added as a region, since the global max and average pooled feature

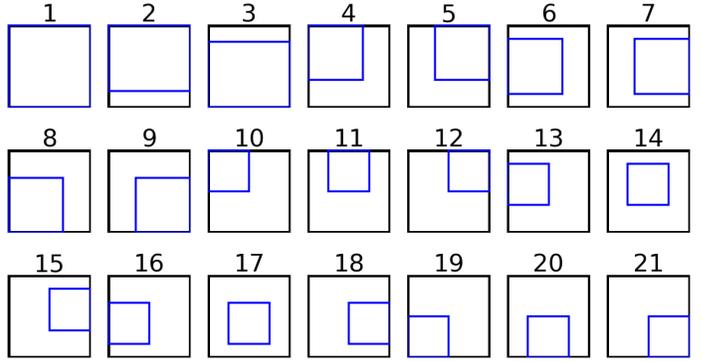


Fig. 3. The 21 regions used for regional max and average pooling

vector yields such good results. This leads to a total of 21 areas (Fig. 3) and 42 pooled values, which will be aggregated in the following steps.

C. Weighted Aggregation of CNN Activations

Performing regional max and average pooling with a grid structure as seen in Fig. 3 on sets of feature maps produced by the last convolutional layer with a ReLU activation, results in a matrix of dimension $42 \times C$ with values in $\mathbb{R}_{\geq 0}$, where C is the number of kernels in the last layer of the chosen CNN. Following the conclusions from the previous experiment, we design a convolutional approach to assign a weight to each of these values (Fig. 4). We refer to this method as DARAC (Deeply Aggregated Regional Activations of Convolutions). In the first step a convolution layer with l kernels of size 42×1 aggregates the pooled data to a matrix of dimension $l \times C$. The output of this operation is followed by a ReLU activation function, which pushes most learned kernel coefficients to be at least 0. The l row vectors of the obtained matrix are further batch normalized [26] to get mean centered feature vectors with unit variance, a representation similar to a PCA whitening output [10] as used in [27] and [6]. Those values are further convoluted with a single $l \times 1$ kernel to output one C -dimensional feature vector. This vector representation is then used as the embedding space to optimize with the described nonlinear rank approximation loss function without any normalization or post processing during the training procedure. The two aggregation layers result in a very small overhead and only add $43l + l + 1$ (incl. bias) learnable parameters to the CNN architecture. We choose $l = 16$ in the following experiments.

After convergence we use a post processing pipeline similar to [27]. Firstly, the extracted feature vector is L^2 normalized and then we perform PCA whitening [10], where the principal components are determined on a subset of the training data set. The whitened vector is subsequently L^2 normalized again.

IV. EXPERIMENTS

A. Training

We chose the ResNet50 v2 architecture [8] as the backbone CNN for the following experiments. Network weights

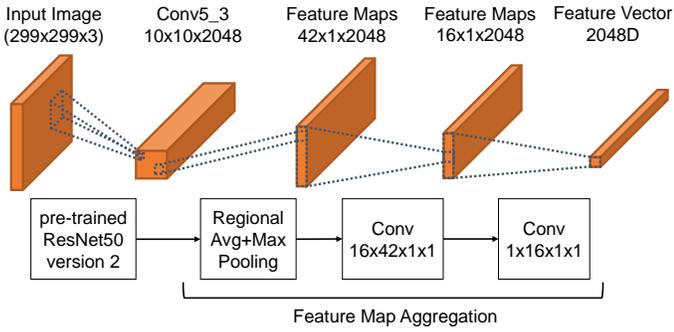


Fig. 4. Functionality of the proposed DARAC System. A two layer convolutional layer with a very small number of trainable parameters is used to aggregate maximum and average activations from different regions of feature maps produced by a CNN.

are initialized with pre-trained parameters achieving a top-1 accuracy of 76.47% on the ImageNet validation data set [19]. The proposed aggregation part is connected to the network’s last convolutional layer with 2048 kernels. The entire system is trained on the *Google Landmarks* data set [15]. We choose $k = 16$ classes and $n = 4$ representatives for batch building. Therefore we build a subset of the landmarks data set, where all images from classes with less than four examples are excluded. This modification leads to a total of 1 196 934 images from 12 602 classes. 100 000 randomly chosen batches are processed in total and we use stochastic gradient descent with momentum as the optimizer with a learning rate of $\alpha = 0.0001$ throughout the entire training process. First all images are resized to 320 pixels on the smaller side and then they are randomly cropped to form a square with 299 pixels. These crops are subsequently randomly flipped horizontally for data augmentation purposes. Since the training on a cleaned landmarks data set appeared to be crucial for optimal retrieval results in [6] we perform an additional training on a subset of the used collection with identical hyper-parameters after convergence. However, only 28 326 images from 582 classes could be retrieved due to broken links. The resulting feature vectors from the first training procedure are further denoted as DARAC-GL and the vectors produced after the second training as DARAC-CL. We use the *TensorFlow* framework [1] for the DARAC implementation and all experiments are conducted on a single GeForce 1080 Ti GPU with 12 GB of memory.

B. Evaluation

Both network states are evaluated on three widely used retrieval data sets and we strictly follow the respective protocols for calculations of the mean average precision scores. The Paris [17] and Oxford Building [16] data sets consist of various landmarks, which makes them similar to the ones used during training. The Holidays data set [9] is more diverse and contains a mix of images other than landmarks. Similar to [18] and [6], we aggregate feature vectors extracted from different image scales. More specifically the images are resized to the resolutions of $S \in \{299, 540, 1020\}$. Each vector is post processed in the previously described manner

TABLE I
MAP SCORES FOR DIFFERENT STAGES OF THE PROPOSED SYSTEM

Method	Image Size	Oxford	Paris	Holidays
ResNet50 (pre-trained)	299	48.7	68.9	86.1
ResNet50-GL (trained)	299	75.3	87.6	91.6
ResNet50-GL (trained)	540	72.5	85.3	90.9
DARAC-GL	299	77.6	89.4	92.8
DARAC-GL + PCA _W	299	81.4	90.8	93.7
DARAC-GL + PCA _W	540	82.2	91.9	95.5
DARAC-GL + PCA _W	1020	78.3	87.6	94.8
MR-DARAC-GL + PCA _W	299, 540, 1020	83.4	93.0	96.9
MR-DARAC-CL + PCA _W	299, 540, 1020	88.2	94.1	95.5

individually and the whitened descriptors are sum aggregated and L^2 normalized subsequently. This approach is denoted as (multi-resolution) MR-DARAC.

C. Results

Table I summarizes the retrieval results for the evaluation data sets for each of the described steps. As a baseline we show mAP scores for the global average pooled feature vector obtained from the pre-trained and trained ResNet50 network. The fine tuning on the Google Landmarks data set increases performance on all data sets but especially on Paris and Oxford Buildings, since those are very close to the training set. However, increasing the image size from 299 to 540 harms the results obtained with this average pooled layer, which corresponds with the previous findings. The introduced aggregation technique seems to be beneficial on its own but improves results even more, when increasing the image size and after a whitening of the feature vectors. Especially the multi-resolution approach seems to work very well on vectors produced by the introduced deep aggregation of regional convolutional activations. The second training procedure on the Landmarks Clean data set leads to improved results for the two buildings data sets but decreases the mean average precision for Holidays. Even though the authors of [6] claim to exclude images of categories that occur in Paris and Oxford, the overall data set and category distribution appears to be closer to these data sets and therefore leads to better results.

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

Table II puts our method in comparison to similar deep learning based image retrieval systems. Training the proposed network architecture on Google Landmarks images leads to state of the art results for the rotated version of the Holidays data set. An additional fine tuning with the Landmarks Clean data set from [6] results in descriptors that outperform the other methods for Holidays and Oxford. Only DIR [6] achieves better results for the Paris data set. However, it should be noted that DIR and GeM [18] use a ResNet101. Furthermore, DIR spends a total of one week in training, whereas our system converges in under 24 hours on similar GPU hardware.

V. CONCLUSION

The presented DARAC system shows a very efficient method for aggregation of CNN features when fine tuned with the NRA loss for large scale image retrieval. Due to

TABLE II
COMPARISON TO SIMILAR CONTENT BASED IMAGE RETRIEVAL SYSTEMS

Method	Matching	Network	Resolutions / Scales	Oxford	Paris	Holidays
DELF [15]	local	ResNet50	7, 0.25 - 2	83.8	85.0	-
GeM [18]	global	ResNet101	5, 0.25 - 1	87.8	92.7	93.9
DIR [6]	global	ResNet101	550, 800, 1050	86.1	94.5	94.8
MR-DARAC-GL (ours)	global	ResNet50	299, 540, 1020	83.4	93.0	96.9
MR-DARAC-CL (ours)	global	ResNet50	299, 540, 1020	88.2	94.1	95.5

its fully convolutional nature it allows training with relatively small images but achieves great results when using larger resolutions during testing. Additionally, it shows the importance of regional average pooling, which has been ignored in previous works regarding deep learning based image retrieval. A weighted aggregation of average and max pooled regional activations in form of convolutional layers leads to state-of-the-art results while reducing the training and feature extraction time significantly, since images of smaller sizes can be used in both stages. At the same time our method introduces a negligible number of trainable parameters compared to the number of weights in the backbone CNN, while also keeping a very low memory footprint.

REFERENCES

- [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- [2] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. *CoRR*, abs/1406.5774, 2014.
- [3] A. Babenko and V. S. Lempitsky. Aggregating deep convolutional features for image retrieval. *CoRR*, abs/1510.07493, 2015.
- [4] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. *CoRR*, abs/1404.1777, 2014.
- [5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.
- [6] A. Gordo, J. Almazn, J. Revaud, and D. Larlus. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254, 2017.
- [7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [8] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016.
- [9] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *ECCV (1)*, volume 5302 of *Lecture Notes in Computer Science*, pages 304–317. Springer, 2008.
- [10] H. Jgou and O. Chum. Negative evidences and co-occurrences in image retrieval: The benefit of pca and whitening. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *ECCV (2)*, volume 7573 of *Lecture Notes in Computer Science*, pages 774–787. Springer, 2012.
- [11] H. Le and A. Borji. What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks? *CoRR*, abs/1705.07049, 2017.
- [12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.
- [13] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, 2003.
- [14] F. Magliani and A. Prati. An accurate retrieval through r-mac+ descriptors for landmark recognition. In L. Esterle, A. Prati, S. Velipasalar, V. M. Brea, and C. Shan, editors, *ICDSC*, pages 6:1–6:6. ACM, 2018.
- [15] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, pages 3476–3485. IEEE Computer Society, 2017.
- [16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*. IEEE Computer Society, 2008.
- [18] F. Radenovic, G. Tolias, and O. Chum. Fine-tuning cnn image retrieval with no human annotation. *CoRR*, abs/1711.02512, 2017.
- [19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.
- [20] K. Schall, K. U. Barthel, N. Hezel, and K. Jung. Deep metric learning using similarities from nonlinear rank approximations. In *Manuscript submitted for publication to MMSP 2019, IEEE 21st International Workshop on Multimedia Signal Processing, 27-29 Sept 2019, Kuala Lumpur, Malaysia*, 2019.
- [21] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Proceedings of the 16th International Conference on Neural Information Processing Systems, NIPS’03*, pages 41–48. Cambridge, MA, USA, 2003. MIT Press.
- [22] O. Seddati, S. Dupont, S. Mahmoudi, and M. Parian. Towards good practices for image retrieval based on cnn features. In *ICCV Workshops*, pages 1246–1255. IEEE Computer Society, 2017.
- [23] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *NIPS*, pages 1849–1857, 2016.
- [24] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *CVPR*, pages 2206–2214. IEEE Computer Society, 2017.
- [25] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012. IEEE Computer Society, 2016.
- [26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [27] G. Tolias, R. Sivic, and H. Jgou. Particular object retrieval with integral max-pooling of cnn activations. In Y. Bengio and Y. LeCun, editors, *ICLR*, 2016.