Local Convolutional Features with Unsupervised Training for Image Retrieval

Mattis Paulin [1] Matthijs Douze[1] Zaid Harchaoui[1,2]
Julien Mairal[1] Florent Perronnin[3] Cordelia Schmid[1]

[1] Inria * [2] NYU [3] Facebook AI Research†

Abstract

Patch-level descriptors underlie several important computer vision tasks, such as stereo-matching or content-based image retrieval. We introduce a deep convolutional architecture that yields patch-level descriptors, as an alternative to the popular SIFT descriptor for image retrieval. The proposed family of descriptors, called Patch-CKN, adapt the recently introduced Convolutional Kernel Network (CKN), an unsupervised framework to learn convolutional architectures. We present a comparison framework to benchmark current deep convolutional approaches along with Patch-CKN for both patch and image retrieval, including our novel "RomePatches" dataset. Patch-CKN descriptors yield competitive results compared to supervised CNN alternatives on patch and image retrieval.

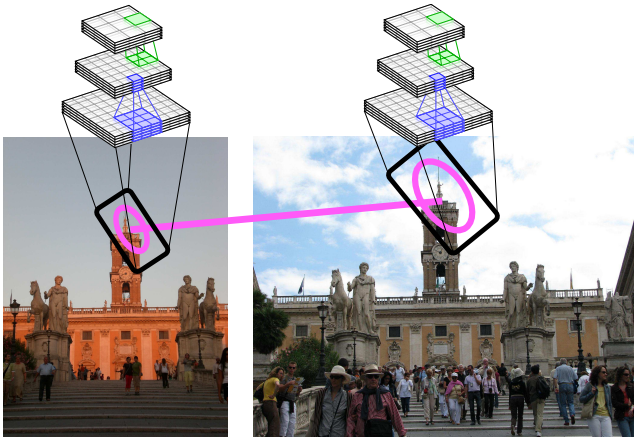

Figure 1. Proposed approach: interest regions are first extracted from images; then the neighborhood of each point is affine- and orientation-normalized to sample a patch; finally, deep convolutional nets are used to get a descriptor for each patch.

1. Introduction

This paper introduces a deep kernel-based convolutional approach for the description of image patches that does not require supervision. The kernel-based feature representation can be effectively approximated using a simple stochastic gradient optimization procedure, yielding a patch-level descriptor that can be used for image retrieval tasks. Image retrieval is a challenging problem as different images of the same object/scene may exhibit large variations in viewpoint, illumination, scaling, occlusion, etc – see Figure 1.

State-of-the-art instance-level retrieval systems involve three steps: 1) *interest point detection*, 2) *description* and 3) *matching*. The goal of step 1) is to select key points that are reproducible under scale and viewpoint changes – see [30, 43] for a detailed comparison of detectors. The choice of a good local representation at step 2) is crucial to ensure robustness to viewing conditions. As an example, the popular SIFT descriptor [26] is robust to illumination variations or to small rotations.[1] As for step 3), the goal is to define a suitable metric between two patch sets. To avoid the cost of matching individual patches, scalable alternatives have been proposed that encode and aggregate local patch statistics, such as bag-of-words [41] and VLAD [19]. In this work, we focus on step 2), *i.e*. the description step, while we rely on state-of-the-art components for the detection and matching steps.

Our inspiration comes from the expressive feature representations output by deep convolutional neural nets (CNNs) [23] used in image classification [22]. Features output by a CNN's intermediate layers can be used as *image-level descriptors* [10] which can be transferred to a variety of tasks – see *e.g*. [3, 13, 31]. More recently, the question of whether suitable *patch-level descriptors* could be derived from such architectures has been raised. [25, 12, 39] provide a preliminary positive answer to this question by comparing favorably with the SIFT descriptor. While these works exhibit significant differences, it is worth noting they

*LEAR team, Inria Grenoble Rhône-Alpes, Laboratoire Jean Kuntzmann, CNRS, Univ. Grenoble Alpes, France.

†Research conducted while at the Xerox Research Centre Europe.

[1] SIFT refers only to the description part.

all rely on *supervised learning* processes.

While the penultimate layer outputs a suitable image-level descriptor [3, 13, 31], the output of previous layers, typically the 4th one, is actually preferable to obtain expressive patch-level descriptors, as noticed in [25]. As shown in [47], earlier layers tend to encode more task-independent information with respect to the later ones. In particular, the filters learned by the first layer tend to be similar regardless of the task, the objective function or the *level of supervision*. This motivates the following question: *is supervised learning required to make good local convolutional features for patch matching and image retrieval?*

Our contribution is the family of patch descriptors Patch-CKN based on Convolutional Kernel Networks (CKNs) [27]. CKNs were initially introduced for image classification in [27]. We introduce a feature representation for patches that is based on the kernel feature map of a convolutional match kernel, and therefore does not depend on examples nor labels. A finite-dimensional explicit feature embedding can be computed to approximate this kernel feature map [34, 44, 27]. We present a fast and simple procedure to compute an explicit feature embedding, using random sub-sampling of the patches, suitable for large-scale experiments. The experiments suggest that Patch-CKN gives competitive patch-level descriptors compared to supervised CNNs.

Several works have focused on learning patch representations [7, 46], yet few have analyzed the impact of improving patch retrieval on image retrieval performance. For this purpose, we introduce a new dataset, "RomePatches", using images and the 3D reconstruction of [1]. The 16K Flickr images of "RomePatches" represent views of 66 different locations in Rome. The 3D reconstruction provides sparse patch matches, yielding the ground truth for our patch retrieval dataset. This allows relating performance improvements for both patch and image retrieval tasks. In a nutshell, our main contributions are three-fold:

1. We propose a patch descriptor based on a CKN architecture [27], using a fast and simple stochastic procedure to compute an explicit feature embedding.

2. We introduce and make available a dataset, named "Rome-Patches", for the evaluation of patch *and* image retrieval, enabling to systematically study the correlation between patch matching and image retrieval performance.

3. We show that, for the purpose of patch and image retrieval, it is possible to learn competitive patch-level descriptors without supervision, and therefore at a fraction of the computational and annotation cost compared to previous supervised alternatives [25, 11].

Overview. We review related work in Sec. 2. In Sec. 3, we describe our image retrieval pipeline, and devote Sec. 4 to convolutional descriptors. Our new dataset is introduced in Sec. 5, and experimental results are presented in Sec. 6. Our new dataset as well as the code to extract Patch-CKN are available online[2].

2. Related Work

Our literature review focuses on the works which are closest to ours: shallow patch descriptors, deep learning for image retrieval and deep learning for patch description.

Traditional patch descriptors. Among the variety of standard patch descriptors, SIFT [26] is the most widely used. Interpreted as a convolutional net, SIFT is a two-layer architecture, the first layer computing patch gradient orientations, average-pooled in the second one. SIFT has been successfully used for many tasks such as stereo matching [1], content-based retrieval [16], or classification [33]. Mikolajczyk *et al*. [29] provide a detailed survey of local descriptors and demonstrate the excellent performance of SIFT. Improved local descriptors include BRIEF [8] and LIOP [45]. All these descriptors are hand-crafted and have been optimized by grid-search on a relatively small amount of parameters. When the number of parameters to be set is large, such as approach is infeasible and the optimal parametrization needs to be learned from data.

Most works on hand-crafted descriptor learning use supervision. Brown *et al*. [7, 46] designed a matching dataset from reprojected 3D models of landmarks, obtained by structure from motion, with a descriptor consisting of several existing parts, including but not limited to SIFT, GLOH [29] and Daisy [42]. We do not include their dataset in our experiments, because of significant differences in early stages of the pipeline as the Multi-view Stereo Correspondence Dataset contains few images of just three locations, with grey-scale patches extracted (while in this work we leverage the additional color information), that were extracted with a detector that is incompatible with ours (DoG instead of Hessian-affine). Philbin *et al*. [37] learn a Mahalanobis metric for SIFT descriptors to compensate the binarization error, with excellent results in instance-based retrieval. Simonyan *et al*. [40] propose the "Pooling Regions" descriptor and learn its parameters, as well as a linear projection using stochastic optimization. Their learning objective can be cast as a convex optimization problem, which is not the case for classical convolutional networks.

An exception is [5] which presents a match-kernel interpretation of SIFT, and a family of kernel descriptors whose parameters are learned in an unsupervised fashion. The Patch-CKN we introduce generalizes kernel descriptors; the proposed procedure for computing an explicit feature embedding is faster and simpler.

[2] lear.inrialpes.fr/people/paulin/projects/RomePatches

descriptor	application	supervision	#parameters		optimization method
SIFT [26]	sparse features	N/A	2	0	N/A
Daisy [46]	patch matching	class = 3D location	0	10	Powel's conjugate direction method
AlexNet [22]	image classification	object classes	50	70M	backpropagation, SGD
Neural codes [3]	same-image recognition	landmark images	50	70M	fine-tuning on top of AlexNet
PhilippNet [12]	patch matching	artificial classes	10	10k	backpropagation, SGD
Fracking [39]	patch matching	match/non-match	10	46k	backpropagation, SGD
CKN [27]	image classification	no supervision	10	256k	layer-wise SGD

Table 1. Levels of supervision and optimization methods used by the approaches related to this work. There are two columns for parameters: hyper-parameters (tuned by hand) and parameters determined by the optimization method.

Deep learning for image retrieval. With a CNN learned on a sufficiently large labeled set such as ImageNet [9], the output of its intermediate layers can be used as image descriptors for a wide variety of tasks including image retrieval [3, 38] – the focus of this work. The output of one of the fully-connected layers is often chosen because it is compact, usually 4,096 D. However, global CNN descriptors lack geometric invariance [14], so they produce results below the state-of-the-art in instance-level image retrieval. Hence, improvements have been proposed.

In [38, 14], CNN responses at different scales and positions are extracted. We proceed similarly, yet we replace the (coarse) dense grid with a patch detector. There are important differences between [38, 14] and our work. While [14, 38] use the output of the penultimate layer as patch descriptor, we show in our experiments that we can get improved results with the output of preceding layers, that are cheaper to compute. In [3], the authors use a single global CNN descriptor for instance-based image retrieval and fine-tune the descriptor on a surrogate landmark dataset. While fine-tuning improves results, it would be difficult to replicate this success beyond landmarks. Finally, [21] proposes a Siamese architecture to train image retrieval descriptors but does not report results on standard retrieval benchmarks.

Deep patch descriptors. Recently [25, 12, 39] reported superior results to SIFT for tasks such as patch matching or patch classification. The three works use different levels of supervision to train a CNN: category labels in [25], surrogate patch labels in [12] (each class is a given patch under different transformations) and matching/non-matching pairs in [39]. There are two key differences between those works and ours. First, they focus on patch-level metrics, instead of actual image retrieval. Second, and more importantly, while all these approaches require some kind of supervision, we show that our Patch-CKN yields competitive performance in both patch matching and image retrieval without requiring supervision. Especially, with respect to [25, 39] we do not need costly labels. And compared to [12] we do not need to make arbitrary choices in the definition of classes (*i.e*. the set of transformations). Table 1 summarizes the competing approaches.

3. Image Retrieval Pipeline

We briefly present the three-step pipeline: interest point detection, patch description, and patch matching.

Interest point detection. Interest point detectors provide locations invariant to certain image transformations. This ensures that two views of the same scene even with changes in viewpoint or illumination share similar "interest points", see [30] for a review of detectors. We use the popular Hessian-Affine detector [28]. The idea is to extract points at their characteristic scale and estimate for each point an affine-invariant local region, see Fig. 1. Rotation invariance is obtained by rotating patches to align the dominant gradient orientation. This results in a set of interest points associated with locally affine-invariant regions.

Interest point description. Given a normalized patch M obtained by mapping the affine region to a fixed-size square, we compute its feature representation $\phi(M)$ in a Euclidean space. The representation is expected to be robust to the perturbations that are not covered by the detector (lighting changes, small rotations, blur,...).

Patch matching. Because matching all possible pairs of patches is too expensive, we follow the standard practice of encoding the patch descriptors and aggregating them into a fixed-length image descriptor, using the VLAD representation [18]. Given a clustering of the feature space consisting of k centroids $\{c_1, \ldots, c_k\}$, VLAD encodes a set of descriptors as the total shift with respect to their assigned centroid. A power normalization with exponent 0.5 is then applied to the VLAD descriptor, as well as an L_2 normalization.

4. Convolutional Descriptors

We use convolutional features to encode fixed-size image patches (size 51×51 pixels). CNNs are normally trained with class supervision for a classification task. This can be extended to image retrieval by either: (i) encoding local descriptors with a model that has been trained for an unrelated image classification task, see Section 4.1; (ii) devising a surrogate classification problem that is as related as possible to image retrieval; (iii) using unsupervised learning, such as a convolutional kernel network, see Sec. 4.2.

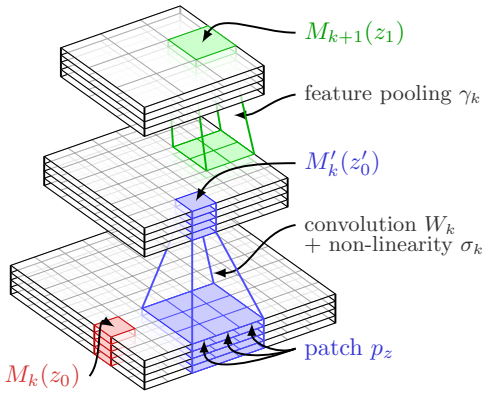

Figure 2. A typical organization for two successive layers of a CNN. The spatial map $M_{k'}$ is obtained from M_k by convolution and pointwise non-linearity, and the top layer M_{k+1} is obtained from M'_k by a downsampling operation called feature pooling. By convention the map M_0 corresponds to the input image x.

4.1. Convolutional Neural Networks

Convolutional neural nets transform an input image by a sequence of simple operations or layers. Each layer performs a linear operation followed by a pointwise non-linearity. Formally, the output $f(x)$ of a CNN for some image x represented as a vector is

$$f(x) = \gamma_K(\sigma_K(W_K \ldots \gamma_2(\sigma_2(W_2\gamma_1(\sigma_1(W_1 x))\ldots)), \quad (1)$$

where the terms W_k are matrices corresponding to linear operations, the functions σ_k are pointwise non-linear functions, e.g., sigmoids or rectified linear units, and the functions γ_k perform a downsampling operation (feature pooling). For a convolutional layer, the matrices W_k have a particular structure and correspond to convolutions of spatial maps, as illustrated in Fig. 2. When they are dense and unstructured, the layer is called "fully connected".

Learning from category labels. The most popular off-the-shelf CNN is AlexNet [22], which won the ImageNet 2012 challenge. AlexNet has 7 layers: the first five are convolutional and the last ones are fully connected. The network is designed to process images of size 224×224, but convolutional layers may be fed with smaller inputs to produce 1x1 maps that we can use as low-dimensional patch descriptors – see the "coverage" column in Table 3. To ensure a fair comparison between all approaches, we rescale patches to always produce a 1×1 map.

Learning from surrogate labels. Most CNNs such as AlexNet augment the dataset with perturbed versions of training patches to learn the filters W_k in (1). The authors of [11, 12] use "virtual patches", obtained as transformations of randomly extracted ones to fall back to a classification problem. For a set of patches $\mathcal{P}$, and a set a transformations $\mathcal{T}$, the dataset consists of all $\tau(p)$, $(\tau, p) \in \mathcal{T} \times \mathcal{P}$. Transformed versions of the same patch share the same label, thus defining surrogate classes. In this paper, we evaluate this strategy by using the same architecture and filter values, called PhilippNet, as in [12]. The network has three convolutional and one fully connected layers, takes as input 64x64 patches, and produces a 512-dimensional output.

4.2. Convolutional Kernels Networks

CKNs have the same architecture as classical CNNs presented in Eq. (1) and in Fig. 2. The feature representation of CNNs relies on filters that are learned and hence defined in a data-dependent manner. We define here a feature representation that is based on a kernel (feature) map. The exact version of this feature representation is therefore data-independent. An explicit kernel (feature) map can be computed [34, 44, 27] to approximate it for computational efficiency. We present here a fast and simple procedure for this purpose, using sub-sampling of patches and stochastic gradient optimization, yielding a CKN that outputs patch descriptors.

Let M and M' be two patches of size $m \times m$ ($m = 51$ in this paper), and $\Omega = \{1, \ldots, m\}^2$ be the set of pixel locations. Let us also consider a fixed sub-patch size and denote by p_z the sub-patch from M (resp. p'_z the sub-patch from M') centered at location $z \in \Omega$.[3]

Single-layer kernel definition. We consider the following kernel [27]:

$$K_1(M, M') = \sum_{z, z' \in \Omega} e^{-\|z - z'\|^2 / 2\beta_1^2} k_1(p_z, p'_{z'}), \quad (2)$$

where

$$k_1(p_z, p'_{z'}) = \|p_z\| \|p'_{z'}\| e^{-\|\tilde{p}_z - \tilde{p}'_{z'}\|^2 / 2\alpha_1^2}, \quad (3)$$

α_1 and β_1 are two kernel hyperparameters, $\|\cdot\|$ denotes the usual L_2 norm, and $\tilde{p}_z$ and $\tilde{p}'_{z'}$ are L_2-normalized versions of the sub-patches p_z and $p'_{z'}$.

The corresponding kernel (feature) map defines a feature representation for patches and images. Furthermore, the kernel is a match kernel. Therefore, the kernel offers a tunable level of invariance through the choice of hyperparameters, and produces hierarchical convolutional representations that are well-suited for natural images.

Kernel embedding approximation. Since the exact computation of (2-3) is overwhelming, Mairal *et al.* propose an explicit finite-dimensional embedding [34, 44] to approximate it. The embedding of [27] keeps the 2-D spatial structure, similar to CNN feature maps. For the one-layer CKN, the approximation of [27] is:

$$K_1(M, M') \approx \sum_{u \in \Omega_1} g_1(u; M)^T g_1(u; M')$$

[3] In practice, sub-patches near the border of M which have values outside of the domain Ω are discarded from the sum (2).

with for all $u \in \Omega_1$,

$$g_1(u; M) := \sum_{z \in \Omega} e^{-\|u-z\|^2/2\beta_1^2} h_1(z; M),$$

and, for all $z \in \Omega$,

$$h_1(z; M) := \|p_z\| \left[\sqrt{\eta_j} e^{-\|w_j - \tilde{p}_z\|^2/\alpha_1^2}\right]_{j=1}^{n_1},$$

where Ω_1 is a subset of Ω as in [27] and w and η are learned parameters. There are two distinct approximations, one in the subsampling defined by $|\Omega_1| \leq |\Omega|$ that corresponds to the stride of a CNN pooling operation, and one in the embedding of the Gaussian kernel of the subpatches: $k_1(p_z, p'_{z'}) \approx h_1(z; M) h_1(z'; M')$.

Since $K_1(M, M')$ is a sum of the match-kernel terms, we can approximate it at sub-patch level by solving an optimization problem. In contrast to the original formulation in Eq. 4 of [27], we introduce the change of variables

$$b_j = \log(\eta_j)/2 - (1 + \|w_j\|^2)/\alpha_1^2$$
$$\omega_j = 2w_j/\alpha_1^2$$

and, considering a sample of n pairs of sub-patches $\{(p_i, p'_i)\}_{i=1,\dots,n}$, we solve:

$$\min_{w_j, \eta_j} \sum_{i=1}^{n} \left(e^{-\frac{\|\tilde{p}_i - \tilde{p}'_i\|^2}{2\alpha_1^2}} - \sum_{j=1}^{n_1} \eta_j e^{-\frac{\|w_j - \tilde{p}_i\|^2}{\alpha_1^2}} e^{-\frac{\|w_j - \tilde{p}'_i\|^2}{\alpha_1^2}} \right)^2$$

We use stochastic gradient optimization to find a stationary point of this (non-convex) objective. This is much faster than the original L-BFGS optimizer [27]; see Sec. 6.

Multi-layer CKN kernel. A kernel can be overlaid on top of the single kernel for a "deeper" and potentially better feature representation [4]. Given an input patch M, the single-layer CKN defines an approximation $f_1(M)$ that can be interpreted as a spatial map. It is possible to define a kernel K_2 on this map in the same way as we have done for input patches. For that, we simply define a patch size, new hyper-parameters β_2 and α_2, and replace M, M' by $f_1(M), f_1(M')$ in all equations of the previous section. Figure 3 gives an illustration of the corresponding two-layer convolutional kernel. Training a multi-layer CKN is naturally sequential, one layer after another.

Input types. We investigate three possible inputs for our CKNs. The first, CKN-raw, directly feeds the raw RGB patch to the network. This scheme captures the hue information, which can prove a drawback in certain situations.

CKN-white consists in pre-processing each sub-patch of the CKN's first layer, by subtracting their mean color, and using PCA-whitening, with a PCA learned on all sub-patches of the initial patch. This responds only to local variations inside the sub-patch, and makes the network more invariant to color.

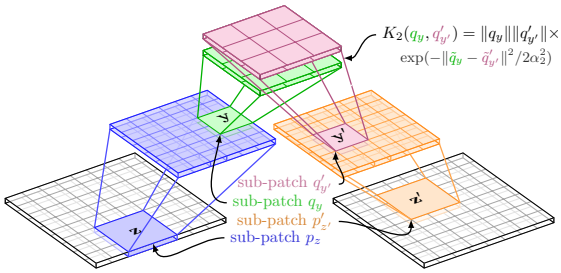

Figure 3. The two-layer convolutional kernel architecture. Each layer is a weighted match kernel between all patches of the previous one; q_z (resp $q'_{z'}$) is a sub-patch of p_z (resp $p'_{z'}$), which is a itself a sub-patch of M (resp M'). The two-layer CKN provides an approximate explicit feature map of this kernel. See [27] for details.

CKN-grad is fully invariant to color. It is the gradient along each spatial dimension with 1×1 sub-patches – that is, the sub-patch $\tilde{p}_z$ for this first layer is simply two-dimensional and can be written $p_z = (G_x, G_y)$. Because the features are normalized, the inner part of the match kernel $\|\tilde{p}_z - \tilde{p}'_{z'}\|$ is directly linked to the cosine of the angle between the two gradients, see [5, 27]. Indeed, an explicit approximation of the kernel K_1 with n_1 evenly distributed orientations $\theta_j = 2j\pi/n_1,\ j \in \{1, \dots, n_1\}$ writes:

$$e^{-\|\tilde{p}_z - \tilde{p}'_{z'}\|^2/2\alpha_1^2} \approx \sum_{j=1}^{n_1} \varphi_1(j; p_z) \varphi_1(j; p'_{z'}),$$

where for all j,

$$\varphi_1(j, p_z) = e^{-\left((\cos\theta_j - G_x/\rho)^2 + (\sin\theta_j - G_y/\rho)^2\right)/\alpha_1^2}$$

and $\rho = \sqrt{G_x + G_y}$. This formulation can be interpreted as a soft-binning of gradient orientations in a "histogram" of size n_1. To ensure an adequate distribution in each bin, we set $\alpha_1 = \left((1 - \cos(2\pi/n_1))^2 + \sin(2\pi/n_1)^2\right)^{1/2}$.

5. Datasets

We conduct experiments for two tasks, patch and image retrieval. We introduce a new dataset for both, which we describe in this section, together with the standard benchmarks.

5.1. Patch retrieval

The Mikolajczyk Dataset. Designed to benchmark interest points detectors and descriptors, the Mikolajczyk dataset [29] contains a set of 8 scenes with 6 images for each. Images of a scene are linked by a homography. We extract regions with the Hessian-Affine detector, and match the corresponding descriptors using Euclidean nearest neighbor. The match between a pair of ellipses is counted correct if the projection of the region with the

ground-truth homography to the second image overlaps by at least 50%. Mean average precision (mAP) is used as performance measure.

RomePatches. Since the existing patch retrieval datasets we are aware of do not contain color information and are not extracted with our detector (Hessian-Affine), we introduce a new dataset[4]. Similar to [46], we use the 3D-reconstruction of landmarks to get different views of the same location. We use the Rome16K dataset [24], which consists of 16,179 images of locations in Rome, downloaded from photo sharing sites. Images are partitioned in 66 "bundles", each one containing a set of viewpoints of a given location in Rome (e.g. "Trevi Fountain"). Within a bundle, consistent camera parameters are available for all images[5]. We match the SIFT descriptors of all images using product quantization [17]. Then we keep only matches that verify the epipolar constraint within a tolerance of 3 pixels. Pairwise point matches are then aggregated greedily to form larger groups of 2D points viewed from several cameras. Groups are merged while the reproduction error from the estimated 3D position is below the 3 pixel threshold. Fig. 4 shows matching patches extracted with this algorithm. We split the dataset into two sets of bundles, the train set with 44 bundles on which we are allowed to learn parameters and tune hyper-parameters. The remaining 22 bundles form the test set. From the train as well as the test set, we select 1,000 3D points that are viewed in at least 10 different images and use one as a query and nine randomly sampled as the targets. Our dataset therefore contains 9,000 target points, and 1,000 queries for the train as well as the test set, i.e., a total of 20,000 patches. We report mean average precision (mAP).

5.2. Image Retrieval

RomePatches-Image. Using the aforementioned bundle split, we select 1,000 query images and 1,000 target images evenly distributed over all bundles for both train and test splits. Two images are considered to match if they come from the same bundle, as illustrated in Fig. 4.

Oxford. The Oxford dataset [35] involves 5,000 images of Oxford landmarks. 11 locations in the city are selected as queries. Each location is represented by 5 bounding boxes each extracted from a different image. Given one of the 55 bounding boxes, the task is to find all images of the same location.

UKbench and Holidays. The University of Kentucky benchmark is a set of 10,200 photos. Each group of 4 images represents the same object. Each image is used as a query in turn. The Holidays dataset contains 1,491 photos of scenes and objects. 500 images are used as queries and the queries are excluded from the datasets.

The standard metrics are mAP for Oxford, Paris and Holidays and 4×recall@4 for UKB.

[4]Available online at http://lear.inrialpes.fr/people/paulin/projects/RomePatches/

[5]http://www.cs.cornell.edu/projects/p2f/

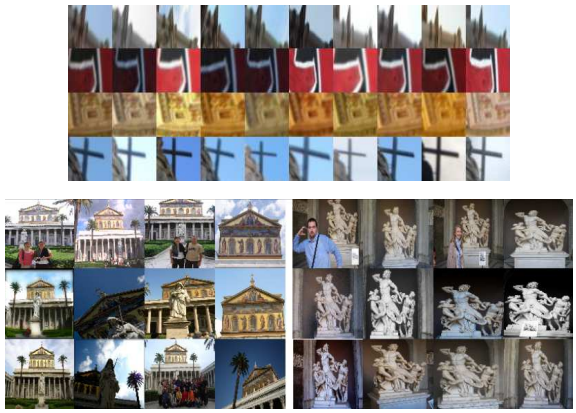

Figure 4. Patch and image retrieval on the Rome dataset. Top: examples of matching patches. Bottom: Images of the same bundle, that therefore share the same class for image retrieval.

6. Experimental Results

After describing implementation details, we report results for patch and image retrieval.

6.1. Implementation details

As our goal is to optimize local descriptors, all methods are given the same patch information as input (computed at Hessian-affine interest points), and are evaluated with the same global descriptor (VLAD with 256 centroids). We believe that improvements in feature detection and aggregation would benefit all architectures equally, without changing the relative performance of patch descriptors.

Patch extraction. As input for all methods, we use 51×51 pixel patches, which was found to be optimal on SIFT descriptors for the Oxford dataset.

CNN implementation. For CNNs, we use the popular Caffe framework [20], and the provided AlexNet (learned on ImageNet 2012). For the PhilippNet [12], we used the model provided by the authors. As explained in section 4, we rescale the 51x51 input patches to the size that, when fed to the CNN, produces 1x1 output maps. Rescaling artifacts do not have a noticeable impact compared to re-extracting patches.

Details of CKN learning. AlexNet and PhilippNet are provided with their parameters, we only learn CKNs. To do so, we randomly select a set of 100K patches in the train split of RomePatches. For each layer, 1 million sub-patches corresponding to convolution areas are extracted and all pairs of patches are fed to the objective function (4.2). The SGD optimization is run for 300K iterations with a batchsize of

1000. Because the objective is nonconvex, several tricks were used, such as random initialization, preconditioning (optimization is conducted in a space where the patch entries are decorrelated), selecting an initial learning rate in the range $\{1, 2^{-1/2}, 2^{-1}, \ldots, 2^{-20}\}$ by performing 1K iterations and choosing the one giving the lowest objective evaluated on a validation set [6]; after choosing the learning rate, we keep monitoring the objective on a validation set every 1K iteration, and perform backtracking in case of divergence. The learning rate is also divided by $\sqrt{2}$ every 50K iterations. These heuristics are fixed over all experiments. Training a CKN takes roughly 10 min on a GPU compared to 2-3 days for the L-BFGS implementation of [27]. As CKN and CNN share the same architecture, the descriptor extraction time is similar for all convolutional methods.

6.2. Patch retrieval

Because the evaluation is computationally cheaper for the patch retrieval task than for image retrieval (10K patches to encode for RomePatches, against more than 4M for Holidays), we optimize the hyperparameters of our CKNs on the RomePatches dataset. We select the best parameters on the train split, without accessing the test data.

Parametric exploration of CKNs. We explore the three input types separately. For each layer, four hyperparameters have to be determined: the size of the convolutional mask (sub-patch size), the coefficient α_k, the pooling factor and the number of outputs (n_k). The spatial comparison coefficient β_k is related to the pooling factor and is set as in [27]—that is, to the pooling factor divided by $\sqrt{2}$. We determine α_k as a quantile σ_k of the distribution of pairwise distances between sub-patches. This value was found optimal at 10^{-3} for all architectures, a much smaller value than reported in [27], which suggests that image classification requires more invariance than patch matching.

As mentioned before, we optimize these parameters over the train split of RomePatches. We try the values 2, 3, 4 and 5 for the sub-patch sizes and pooling factors with 128, 256, 512 or 1024 outputs. The α parameter was selected in the $\{0.1, 0.01, 0.001\}$ quantiles. The retained parameters are given in Table 2. To the notable exception of color, architectures perform better with two layers. In general, the higher the number of features, the better performance.

Input	Layer 1	Layer 2	dim.
CKN-raw	5x5, 5, 512	—	41472
CKN-white	3x3, 3, 512	2x2, 2, 512	32768
CKN-grad	1x1, 3, 16	4x4,2,1024	50176

Table 2. For each layer we indicate the sub-patch size, the subsampling factor and the number of filters. For the gradient network, the value 16 corresponds to the number of orientations.

In the following, we use the best architectures given in Table 2 for each input type.

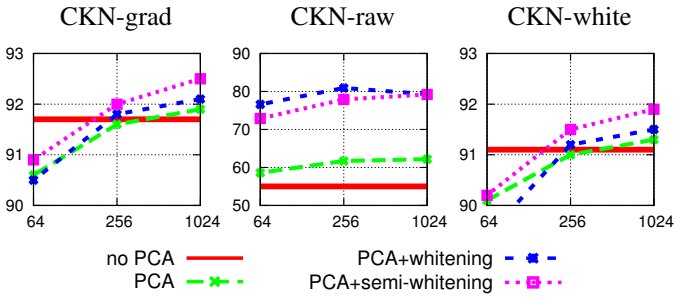

Figure 5. Influence of dimensionality reduction on patch retrieval performance. Results reported in mAP (%) on the train split of RomePatches as a function of the PCA dimension. As a comparison, SIFT reports 91.6%.

Architecture	coverage	Dim	RomePatches train	RomePatches test	Miko.
SIFT	51x51	128	91.6	87.9	57.8
AlexNet-conv1	11x11	96	66.4	65.0	40.9
AlexNet-conv2	51x51	256	73.8	69.9	46.4
AlexNet-conv3	99x99	384	81.6	79.2	53.7
AlexNet-conv4	131x131	384	78.4	75.7	43.4
AlexNet-conv5	163x163	256	53.9	49.6	24.4
PhilippNet	64x64	512	86.1	81.4	59.7
CKN-grad	51x51	1024	**92.5**	**88.1**	59.5
CKN-raw	51x51	1024	79.3	76.3	50.9
CKN-white	51x51	1024	91.9	87.7	**62.5**

Table 3. Results of convolutional architectures for patch retrieval.

Comparative results. We compare the convolutional architectures on our three patch datasets: RomePatches-train, RomePatches-test and Mikolajczyk. Results are given in Table 3. For AlexNet CNNs, we report results for all outputs of the 5 convolutional layers (after ReLU). We note that SIFT is an excellent baseline for these methods, and that CNN architectures that were designed for local invariances perform better than the ones used in AlexNet, as observed in [12]. The results of the PhilippNet on the Mikolajczyk dataset are different from the ones reported in [12], for several reasons. First, we evaluate on Hessian-Affine descriptors while they use MSER. To have a comparable setting, we use their network with an input of 64x64, while they slide it on 91x91 patches. Such an additional layer results in a small increase of performance (2% for patch retrieval and 1% for image retrieval). We observe that PhilippNet outperforms both SIFT and AlexNet, which was the conclusion of [12]; CKN trained on whitened patches do however yield better results.

6.3. Image Retrieval

Settings. We learn a vocabulary of 256 centroids on a related database: for Holidays and UKB we use 5000 Flickr images and for Oxford, we train on Paris [36]. For

	Holidays	UKB	Oxford	Rome	
				train	test
SIFT	64.0	3.44	43.7	52.9	62.7
AlexNet-conv1	59.0	3.33	18.8	28.9	36.8
AlexNet-conv2	62.7	3.19	12.5	36.1	21.0
AlexNet-conv3	**79.3**	3.74	33.3	47.1	54.7
AlexNet-conv4	77.1	3.73	34.3	47.9	55.4
AlexNet-conv5	75.3	3.69	33.4	45.7	53.1
PhilippNet	74.1	3.66	38.3	50.2	60.4
CKN-grad	66.5	3.42	**49.8**	**57.0**	**66.2**
CKN-raw	69.9	3.54	23.0	33.0	43.8
CKN-white	78.7	3.74	41.8	51.9	62.4
CKN-mix	**79.3**	**3.76**	43.4	54.5	65.3

Table 4. Image retrieval results. CKN-mix is the result of the concatenation of the VLAD descriptors for the three channels.

RomePatches-Train and RomePatches-Test the vocabulary is learned the other one. The final VLAD descriptor size is 256 times the local descriptor dimension.

Comparative results. We compare all convolutional approaches as well as the SIFT baseline in the image retrieval settings. Results are summarized in Table 4.

On datasets for which color is dominant (e.g. Holidays or UKB), the best individual CKN results are attained by CKN-white, improved by combining the three channels. On images of buildings, gradients still perform best and the addition of color channels is harmful, which also explains the poor performance of AlexNet. On the other hand, PhilippNet was trained to be invariant to colorimetric transformations, and therefore yields better results than its CNN counterpart.

Comparison with the state of the art. Table 5 compares our approach to recently published results. Approaches based on VLAD with SIFT [2, 19] can be improved significantly by CKN local descriptors (+15% on Holidays). To compare to the state of the art with SIFT on Oxford [2], we use the same Hessian-Affine patches extracted with gravity assumption [32]. Note that this alone results in a 7% gain.

We also compare with global CNNs [3]. Our approach outperforms it on Oxford and UKB and is on par on Holidays. On Holidays, our approach is slightly below the one of [14], that uses AlexNet descriptors and VLAD pooling on large, densely extracted patches. Note that they perform dimensionality reduction and whitening, which results in a 2% improvement. We plan to investigate dimensionality reduction methods [3, 15] as well as quantization [17] in future work.

7. Conclusion

We propose a new descriptor Patch-CKN for patch and image retrieval, that performs on par or better than supervised CNNs on standard patch and image retrieval benchmark datasets and on the proposed RomePatches benchmark dataset.

Method \ Dataset	Holidays	UKB	Oxford
VLAD [19]	63.4	3.47	-
VLAD++ [2]	64.6	-	55.5*
Global-CNN [3]	79.3	3.56	54.5
MOP-CNN [14]	**80.2**	-	-
Ours	79.3	**3.76**	49.8 (**56.5***)

Table 5. Comparison with state-of-the-art image retrieval results. Results with * use a Hessian-Affine detector with gravity assumption [32].

Acknowledgements. This work was partially supported by projects "Allegro" (ERC), "Titan" (CNRS-Mastodons), "Macaron" (ANR-14-CE23-0003-01), the Moore-Sloan Data Science Environment at NYU and a Xerox Research Center Europe collaboration contract.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a day. *Communications of the ACM*, 2011. 2

[2] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013. 8

[3] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 1, 2, 3, 8

[4] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *CVPR*, 2011. 5

[5] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *NIPS*, 2010. 2, 5

[6] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*. Springer, 2012. 7

[7] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *PAMI*, 2011. 2

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *ECCV*, 2010. 2

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 1

[11] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *NIPS*, 2014. 2, 4

[12] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. arXiv Preprint, 2014. 1, 3, 4, 6, 7

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[14] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 3, 8

[15] H. Jégou and O. Chum. Negative evidences and cooccurrences in image retrieval: the benefit of PCA and whitening. In *ECCV*, 2012. 8

[16] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*. 2008. 2

[17] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 2011. 6, 8

[18] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 3

[19] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2012. 1, 8

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. 2014. 6

[21] J. Jiang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 3

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 3, 4

[23] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *NIPS*, 1989. 1

[24] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*. 2010. 6

[25] J. Long, N. Zhang, and T. Darrell. Do Convnets learn correspondances? In *NIPS*, 2014. 1, 2, 3

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1, 2, 3

[27] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *NIPS*, 2014. 2, 3, 4, 5, 7

[28] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 2004. 3

[29] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 2005. 2, 5

[30] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 2005. 1, 3

[31] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. 1, 2

[32] M. Perdoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *CVPR*, 2009. 8

[33] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 2

[34] F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, 2010. 2, 4

[35] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007. 6

[36] 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*, 2008. 8

[37] J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *ECCV*. 2010. 2

[38] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *arXiv Preprint*, 2014. 3

[39] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer. Fracking deep convolutional image descriptors. *Arxiv preprint*, 2015. 1, 3

[40] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 2014. 2

[41] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 1

[42] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *PAMI*, 2010. 2

[43] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 2008. 1

[44] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *TPAMI*, 2012. 2, 4

[45] Z. Wang, B. Fan, and F. Wu. Local intensity order pattern for feature description. In *ICCV*, 2011. 2

[46] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *CVPR*, 2009. 2, 3, 6

[47] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 2