View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

Kernel Square-Loss Exemplar Machines for Image Retrieval

Rafael Rezende, Joaquin Zepeda, Jean Ponce, Francis Bach, Patrick Pérez

To cite this version:

Rafael Rezende, Joaquin Zepeda, Jean Ponce, Francis Bach, Patrick Pérez. Kernel Square-Loss Exemplar Machines for Image Retrieval. Computer Vision and Pattern Recognition 2017, Jul 2017, Honolulu, United States. hal-01515224

HAL Id: hal-01515224

https://hal.inria.fr/hal-01515224

Submitted on 27 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Kernel Square-Loss Exemplar Machines for Image Retrieval

Rafael S. Rezende[1] Joaquin Zepeda[2,3] Jean Ponce[1,4] Francis Bach[1] Patrick Pérez[2]

[1]Inria Paris† [2]Technicolor [3]Amazon‡

[4] École Normale Supérieure/PSL Research University

Abstract

Zepeda and Pérez [41] have recently demonstrated the promise of the exemplar SVM (ESVM) as a feature encoder for image retrieval. This paper extends this approach in several directions: We first show that replacing the hinge loss by the square loss in the ESVM cost function significantly reduces encoding time with negligible effect on accuracy. We call this model square-loss exemplar machine, or SLEM. We then introduce a kernelized SLEM which can be implemented efficiently through low-rank matrix decomposition, and displays improved performance. Both SLEM variants exploit the fact that the negative examples are fixed, so most of the SLEM computational complexity is relegated to an offline process independent of the positive examples. Our experiments establish the performance and computational advantages of our approach using a large array of base features and standard image retrieval datasets.

1. Introduction

The exemplar support vector machine (ESVM), originally proposed by Malisiewicz *et al.* [24], leverages the availability of large, unannotated pools of images within the context of supervised learning. It uses a large generic pool of images as a set of negative examples, while using a single image (the *exemplar*) as a positive example. Given these training sets, an SVM classifier is learned that can generalize well, despite the drastically limited size of the set of positive examples. This classifier has successfully been used in classification, object detection and label transfer [25]. Zepeda and Pérez [41] have proposed to treat instead the weights of the resulting classifier as a new feature vector for image retrieval. An ESVM feature is computed for each database and query image, by treating it as the only positive sample while keeping a fixed pool of generic negative images. Searching amounts to computing distances between the query and database ESVM features. Note that ESVM features can be derived from arbitrary *base* features (*e.g.*, CNN activations) of the exemplar and the images in the generic negative pool.

One drawback of the ESVM feature encoding approach is that computing the classifier requires solving an optimization problem for each positive example (*i.e.*, each query and each database image). This can be time consuming for the large negative pool sizes required for good ESVM feature performance. In this work, we propose using the square loss instead of the hinge loss, in effect converting the ESVM problem into a ridge regression, one that can be solved in closed form. We dub the corresponding classifier a *square-loss exemplar machine* (or *SLEM*). The square loss has been used before to replace the hinge loss in classification tasks (e.g., [37, 40]), and to compare ESVMs to classical classifiers such as the linear discriminant analysis (*LDA*) [21]. In contrast, we propose here to use SLEMs as feature encoders for image retrieval.

Since computing the SLEM features requires inverting a large matrix related to the training set's covariance matrix, we propose an efficient way to compute this inverse. Similarly to the cross-validation method of residual error of [9], we exploit the fact that only a single (positive) example changes in the training set when computing SLEM features for different images. We show experimentally that our representation matches and even improves upon the performance of ESVM features on three standard datasets using a wide range of base features at a fraction of the original computational cost.

We also introduce a kernelized variant of SLEM that enjoys similar computational advantages and improves retrieval performances. Further computational and storage efficiency is obtained using low-rank factorization methods to decompose the kernel matrix of negative samples. We claim this kernelized descriptor and its efficient calculation as the main contribution of this work.

The rest of this paper is organized as follows: In Section

† WILLOW and SIERRA project team, Département d'Informatique de l'École Normale Suprérieure, ENS/Inria/CNRS UMR 8548

2 we provide an overview of various existing feature representation methods. In Section 3 we first review the original ESVM feature representation method and introduce the proposed linear SLEM model. We then introduce the kernel SLEM in Section 4 and present the low-rank approximation that enables its efficient implementation in Section 5. We evaluate the proposed method in image retrieval in Section 6, and present conclusions in Section 7.

2. Prior work

This paper addresses the problem of designing an image representation suitable for content-based retrieval, in particular supporting effective (discriminative) and efficient (fast) comparisons between a query picture and images stored in some large database. These representations must be robust to large image variations due to camera pose, color differences and scene illumination, amongst others.

Many successful approaches to image retrieval rely on unsupervised models of codebook learning, such as K-means [10] or Gaussian mixtures [28, 33]. These approaches aggregate local descriptors of an image by weighted average [27], triangular embedding [18] or generalized max-pooling [26] into a global feature descriptor. Before the neural networks *renaissance*, these representations usually outperformed methods that exploit supervised learning of image features directly [7, 32].

Today, with the success of convolutional architectures, global image descriptors are often obtained by aggregating and/or pooling their last convolutional layers [3, 20, 31] or by addition of new differentiable layers to an existing architecture [1, 14].

3. The square-loss exemplar machine

In this section, we revisit the exemplar SVM model proposed in [24] as an instance of a more general family of classifiers. Then, we introduce the square loss exemplar machine (SLEM) as a simple variant of this model and study its properties.

3.1. Exemplar classifiers

We are given base features in $\mathbb{R}^d$ at training time, one positive example x_0 in $\mathbb{R}^d$ and a set of negative examples $X = [x_1, x_2, ..., x_n]$ in $\mathbb{R}^{d\times n}$, each column of X representing one example by a vector in $\mathbb{R}^d$. We are also given a loss function $l : \{-1, 1\} \times \mathbb{R} \to \mathbb{R}^+$. Learning an exemplar classifier from these examples amounts to minimizing the function

$$J(\omega, \nu) = \theta\, l(1, \omega^T x_0 + \nu) + \frac{1}{n}\sum_{i=1}^{n} l(-1, \omega^T x_i + \nu) + \frac{\lambda}{2}\|\omega\|^2, \tag{1}$$

w.r.t. ω in $\mathbb{R}^d$ and ν in $\mathbb{R}$. In Eq. (1), λ and θ are respectively a regularization parameter on ω and a positive scalar adjusting the weight of the positive exemplar.

Given a cost l, we define the corresponding *exemplar classifiers* of x_0 with respect to X as the weights $\omega^\star(x_0, X)$ that minimizes the loss function J:

$$\left(\omega^\star, \nu^\star\right) = \underset{(\omega,\nu)\in\mathbb{R}^d\times\mathbb{R}}{\operatorname{argmin}} \; J(\omega, \nu).^1 \tag{2}$$

The exemplar SVM [24, 25] is an instance of this model where l is the hinge loss, which is convex. The solution of Eq. (2) can thus be found by stochastic gradient descent [8] individually for each positive sample. The next section shows how to calculate all exemplar classifiers simultaneously by changing the loss function.

3.2. The square loss

Now, let us study the same learning problem for the square-loss function $l(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$. As in the case of the hinge loss, the minimization of Eq. (1) is a convex problem. However it is now a ridge regression problem, whose unique solution can be found in closed form as

$$\begin{cases} \omega^\star &= \dfrac{2\theta}{\theta+1} U^{-1}(x_0 - \mu), \\ \nu^\star &= \dfrac{\theta-1}{\theta+1} - \dfrac{1}{\theta+1}(\theta x_0 + \mu)^T \omega^\star, \end{cases} \tag{3}$$

where:

$$\begin{cases} \mu = & \frac{1}{n}\sum_{i=1}^{n} x_i, \\ U = & \frac{1}{n} X X^T - \mu\mu^T \\ & + \frac{\theta}{\theta+1}(x_0 - \mu)(x_0 - \mu)^T + \lambda \mathrm{Id}_d, \end{cases} \tag{4}$$

where Id_d is the identity matrix of size d.

Woodbury identity. We can simplify Eq. (3) by modifying U in Eq. (4). Let us define $A = \frac{1}{n} X X^T - \mu\mu^T + \lambda \mathrm{Id}_d$ as the regularized covariance matrix and assume its inverse A^{-1} known. The matrix U now reads $U = A + \frac{\theta}{\theta+1}\delta\delta^T$, where $\delta = x_0 - \mu$ is the centered (w.r.t. the negatives' mean) positive sample. The Woodbury identity [39] gives us

$$U^{-1} = A^{-1} - \frac{\theta}{\theta\delta^T A^{-1}\delta + \theta + 1} A^{-1}\delta^T\delta A^{-1}. \tag{5}$$

Substituting (5) in (3) yields

$$\begin{aligned} \omega^\star &= \frac{2\theta}{\theta+1}\left(A^{-1}\delta - \frac{\theta}{\theta\delta^T A^{-1}\delta + \theta + 1} A^{-1}\delta(\delta^T A^{-1}\delta)\right) \\ &= \frac{2\theta}{\theta\delta^T A^{-1}\delta + \theta + 1} A^{-1}\delta. \end{aligned} \tag{6}$$

[1] Depending on the loss function l, $\nu^\star(x_0, X)$ may not be unique.

Equation (6) shows how to compute many exemplar classifiers simultaneously, by solving a single linear system in A. Also note that the positive sample weight θ does not influence the direction of the optimal vector $\omega^\star$, only its norm. This means that if search and ranking are based on the normalized feature $\frac{1}{\|\omega^\star\|}\omega^\star$, *e.g.* using cosine similarity, θ does not influence the matching score of the SLEM vectors of two different images. This sets SLEM appart from ESVM which requires this parameter to be calibrated [24, 41]. We can thus set the value of θ to any positive real number.

3.3. LDA and SLEM

It is interesting to note the relationship between SLEM and the classical linear discriminant analysis (LDA). Let us return to Eq. (1) and suppose that we have multiple positive samples. It can be shown that in this case, the corresponding linear classifier of Eq. (1) for the square loss is also given by (3), where x_0 denotes this time the center of mass of the positive samples *if* the positives have the *same* covariance matrix Σ as the negative samples X.

This equal-covariance assumption is of course quite restrictive, and probably unrealistic in general. It is interesting to note, however, that this is exactly the assumption made by linear discriminant analysis. As shown in [16] for example, LDA is a (non-regularized) linear classifier with decision function $\omega^T x + \nu$, where

$$\begin{cases} \omega = \Sigma^{-1}(x_0 - \mu), \\ \nu = -\dfrac{1}{2}(x_0 + \mu)^T \omega. \end{cases} \tag{7}$$

This shows that, for a single positive sample, SLEM and LDA are very similar: Indeed, taking $\lambda = 0$ (*i.e.* no regularization) and $\theta = 1$, we have $\nu^\star = \nu$, $A = \Sigma$ and that the vectors ω of Eq. (7) and $\omega^\star$ of Eq. (6) have the same direction, reducing SLEM to LDA. Many interesting properties of LDA have been used recently for classification tasks [12, 15]. With our simple generalization of LDA, we hope to obtain superior results.

4. The kernel SLEM

4.1. Kernel methods

Let us recall a few basic facts about kernel methods for supervised classification. We consider a reproducing kernel Hilbert space (RKHS) H formed by real functions over some set X, and denote by k and φ the corresponding reproducing kernel and feature map (which may not admit a known explicit form) over X, respectively. We address the following learning problem over $H \times \mathbb{R}$:

$$\min_{h \in H, \nu \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, \langle \varphi(x_i), h \rangle + \nu) + \frac{\lambda}{2} \|h\|_H^2, \tag{8}$$

where the pairs (x_i, y_i) in $X \times \{-1, 1\}$, $i = 1 \dots n$ are training samples, and $\langle h, h' \rangle$ is the inner product of element h and h' in H. We dub problems with the general form of (8) *affine* supervised learning problems since, given some fixed element h of H and some scalar ν, $\langle h, h' \rangle + \nu$ is an affine function of h', whose zero set defines an affine hyperplane of H considered itself as an affine space.

Let K denote the kernel matrix with entries $k_{ij} = \langle \varphi(x_i), \varphi(x_j) \rangle$ and rows $k_i^T = [k_{i1}, k_{i2}, ..., k_{in}]$, i in $\{1, \dots, n\}$. We assume from now on that l is convex and continuous. Under this assumption, Eq. (8) admits an equivalent formulation

$$\min_{\alpha \in \mathbb{R}^n,\ \nu \in \mathbb{R}} \left(\frac{1}{n} \sum_{i=1}^{n} l(y_i, k_i^T \alpha + \nu) + \frac{\lambda}{2} \alpha^T K \alpha \right), \tag{9}$$

and any solution $(\alpha^\star, \nu^\star)$ to (9) provides a solution $(h^\star, \nu^\star)$ to (8) with $h^\star = \sum_{i=1}^n \alpha_i^\star \varphi(x_i) + \nu^\star$. This result follows from the Riesz representation theorem [34, 38].

Assuming our reproducing kernel is semidefinite positive, K is a semidefinite positive matrix and can be decomposed as $K = BB^T$. Using this factorization, the kernelized problem can be expressed as

$$\min_{\beta \in \mathbb{R}^r, \nu \in \mathbb{R}} \left(\frac{1}{n} \sum_{i=1}^{n} l(y_i, b_i^T \beta + \nu) + \frac{\lambda}{2} \|\beta\|^2 \right), \tag{10}$$

where b_i^T denotes the i-th row of B and r is the number of columns of B. If $(\beta^\star, \nu^\star)$ is the solution of Eq. (10), the corresponding vector $\alpha^\star$ (or, more correctly, *a* corresponding vector of dimension $n \geq r$) can be computed by $\alpha^\star = P\beta^\star$, where P is the pseudoinverse of B^T.

Note that Eq. (10) is written as the usual form of a linear classifier. In particular, it allows us to write the kernel learning problem (8) as an instance of Eq. (1) by setting $\theta = \frac{1}{n}$ (we set this value of θ for the remaining of this work) and $y_i = -1$ for all but one training sample. For our approach, we wish to solve (10) for many positive exemplars against the same set of negative training samples. In the following subsections, we show how to take advantage of the fixed negative samples to efficiently solve (10).

4.2. Offline preprocessing of negative samples

Let us now return to the (kernelized) SLEM, taking l as the square loss. In order to calculate offline all operations that are dependent only on negative samples, let us denote by K the kernel matrix of the negative samples X. The preprocessing phase consists of the calculation of the decomposition B and the constants of Eq. (6): $\mu = \frac{1}{n} \sum_{i=1}^n b_i^T$ and $A = \frac{1}{n} B^T B - \mu\mu^T + \lambda \mathrm{Id}_r$. These operations are done offline and their results are stored.

4.3. Online addition of a positive sample

We now wish to write Eq. (10) as an exemplar classifier, with one positive example x_0 and n negative examples X. We denote by K' the augmented kernel matrix obtained by adding this sample,

$$K' = \begin{bmatrix} k_{00} & k_0^T \\ k_0 & K \end{bmatrix}, \tag{11}$$

where $k_{00} = \langle \varphi(x_0), \varphi(x_0) \rangle$ is a scalar and $k_0 = [\langle \varphi(x_0), \varphi(x_i) \rangle]_{1 \leq i \leq n}$ is a vector in $\mathbb{R}^n$. The following lemma shows how the factorization of K' can be derived from the factorization of its sub-matrix K and the solution of a $n \times n$ linear system.

Lemma 1. *The augmented kernel matrix K' can be factorized as $K' = B'B'^T$ with*

$$B' = \begin{bmatrix} u & v^T \\ 0 & B \end{bmatrix}, \; v = B^\dagger k_0, \; u = \sqrt{k_{00} - ||v||^2}, \tag{12}$$

where $B^\dagger$ is the pseudoinverse of B.

Proof. For B' defined by (12), we have that

$$B'B'^T = \begin{bmatrix} u^2 + \|v\|^2 & v^T B^T \\ Bv & BB^T \end{bmatrix} = \begin{bmatrix} k_{00} & v^T B^T \\ Bv & K \end{bmatrix}. \tag{13}$$

Since K' is positive semidefinite, k_0 must lie in the column space $\mathcal{B}$ of B. Indeed, if we suppose k_0 does not belong to $\mathcal{B}$, then it can be decomposed uniquely as $k_0 = s + t$, $s \in \mathcal{B}$ and $t \in \mathcal{B}^\perp$, with $t \neq 0$. In one hand, K' being semidefinite positive implies that $[1, -at^T]K'[1; -at] = k_{00} - 2a\|t\|^2 \geq 0$[2] for all real value a. In the other hand, for a large enough, $k_{00} - a\|t\|^2 \leq 0$, which is a contradiction. Hence $v = B^\dagger k_0$ is an exact solution of $Bv = k_0$. The fact that $k_{00} - \|v\|^2$ is non-negative comes from the fact that the Schur complement $K - k_0 k_0^T / k_{00}$ of k_{00} in K' is itself positive semidefinite. Indeed, since the matrix $k_{00}K - k_0 k_0^T = B(k_{00}\mathrm{Id}_r - vv^T)B^T$ is also positive semidefinite. Thus $v^T(k_{00}\mathrm{Id}_r - vv^T)v = \|v\|^2(k_{00} - \|v\|^2) \geq 0$. □

This lemma allows us to add a positive sample to Eq. (10). With a positive exemplar, it now reads

$$\frac{1}{n}(b_0'^T \beta + \nu - 1)^2 + \frac{1}{n}\sum_{i=1}^{n}(b_i'^T \beta + \nu + 1)^2 + \frac{\lambda}{2}\|\beta\|^2, \tag{14}$$

with $b_i'^T$ being the $(i+1)$-th row of B', i in $\{0, 1, ..., n\}$. In particular, $b_0' = [u;\ v]$ and, for $i > 0$, $b_i' = [0;\ b_i]$. The solution $(\beta^\star, \nu^\star)$ in $\mathbb{R}^{r+1} \times \mathbb{R}$ can be computed just as before by Eq. (3), replacing x_0 by b_0', μ by $\mu' = \frac{1}{n}\sum_{i=1}^{n} b_i'$ and X by the $(r+1) \times n$ matrix Q of columns $b_1', b_2', ..., b_n'$. The solution $\alpha^\star$ is now calculated as $\alpha^\star = P'\beta^\star$, where $P' = [u^{-1}\ 0^T; -u^{-1}Pv\ P]$ is the pseudoinverse of B'^T. $\alpha^\star$ can be expressed by the linear system

$$\begin{bmatrix} \alpha_0 \\ \hat{\alpha} \end{bmatrix} = \begin{bmatrix} \frac{1}{u} & 0^T \\ -\frac{1}{u}Pv & P \end{bmatrix} \begin{bmatrix} \beta_0 \\ \hat{\beta} \end{bmatrix}. \tag{15}$$

4.4. Similarity score

Once the optimal parameters (β, ν) from (14) and the coordinates u, v of b_0' from (12) have been found[3], they can be used directly for measuring similarity between matching images.

Suppose two image descriptors x_0 and x_0' are given and we wish to calculate the similarity score between their SLEM representations h and h', denoted by $s(h, h')$. We write $h' = \alpha_0'\varphi(x_0') + \sum_{i=1}^{n} \alpha_i'\varphi(x_i) + \nu'$. Using Eq. (15) and ignoring biases ν and ν' which have empirically no influence, $s(h, h')$ is given by:

$$\begin{aligned} s(h, h') &= \langle h, h' \rangle \\ &= \hat{\alpha}^T K \hat{\alpha}' + \alpha_0 k(X, x_0)^T \hat{\alpha}' + \alpha_0' k(X, x_0')^T \hat{\alpha} \\ &\quad + \alpha_0 \alpha_0' k(x_0, x_0') \\ &= \hat{\beta}^T \hat{\beta}' + \lambda^{-2}(k(x_0, x_0') - v^T v). \end{aligned} \tag{16}$$

For a given image whose descriptor is x_0, we need to store x_0, $\hat{\beta}$ and v to calculate its similarity score to whichever other image for SLEM. Since we assume the base feature x_0 has dimension d and $\hat{\beta}$ and v each have dimension r, we store a vector of dimension $d + 2r$ for each image.

5. Efficient implementation

When compared to the linear square-loss classifier of Section 3.2, one drawback of the kernelized approach is that the dimension of our problem grows with the number n of negative samples. The offline factorization BB^T of K demands $O(nr)$ storage and at best $O(nr^2)$ time. This factorization can be obtained in two ways: full-rank and low-rank decomposition. In this section we propose three different decompositions of K and discuss their respective merits.

5.1. Full-rank decomposition

CCD: The complete Cholesky decomposition (CCD) is the most used factorization of positive-definite matrices in kernel-based learning due to its time efficiency [5]. We use it as our default decomposition. We make sure K is positive-definite by adding ϵ to its diagonal, where $\epsilon = \min(0, -\lambda_{min})$ and λ_{min} is the smallest eigenvalue of K. Therefore, B also has rank n and can be calculated by CCD from the identity $BB^T = K + \epsilon\mathrm{Id}_n$.

[2] We use matlab notation for horizontal and vertical staking.

[3] We drop the "$\star$" in this subsection to avoid cluttering the notation.

5.2. Low-rank decomposition

One of the major constraints of large scale retrieval is the minimization of storage. As discussed in Section 4.4, for each database image we store its base representation plus a $2r$ vector. Hence, we aim to decompose K at a small rank r. Two classical methods can be used to obtain a low-rank decomposition of K.

ICD: The incomplete Cholesky decomposition (ICD) is widely used in machine learning [5, 11]. It is similar to CCD, and greedily chooses which column of K to add to the decomposition based on the gain in approximation error [6]. The algorithm stops after r steps, obtaining the factor B in time $O(nr^2)$.

KPCA: Kernel PCA (KPCA) [35] computes the factor B by performing a singular value decomposition of K (truncated singular value decomposition for very small values of r), and making each column of the factor correspond to one of the top r singular vectors. The resulting matrix B is guaranteed to be the best r-rank approximation of K according to the Frobenius norm. The computational cost of KPCA is, however, $O(n^2r)$ [13].

When comparing computation time, KPCA is slower than ICD for small values of r and faster for values of r such that the residue is small. Also, as discussed above, KPCA gives a smaller residue $\mathrm{tr}(K - BB^T)/\mathrm{tr}(K)$. From these comparisons, we set KPCA as our default low-rank decomposition. The only case ICD is more appropriated than KPCA is for very large number of negatives n, for which the time complexity of KPCA becomes an issue, and very small rank r. This particular case is further studied in Section 6.5.

6. Experimental Evaluation

6.1. Datasets and evaluation protocol

We perform experiments on three standard datasets for image retrieval.

- The INRIA *Holidays* dataset [17] consists of 1491 images divided in 500 groups of matching images. We manually rotate by 90 degrees some images that are not in their natural orientation to compensate for the fact that CNN features are not rotation invariant [1, 4, 14, 20, 31].
- The *Oxford5k* dataset [29] consists of 5063 images separated in 55 groups of matching images, each group associated to a landmark of Oxford. We use the "full" crop, ignoring the region of interest of each image.
- The *Oxford105k* dataset [29] is a large-scale dataset containing the same images and queries from Oxford5k plus *Flickr100k*, a collection of 10^5 distractor Flickr images.

As pool of negative images to build SLEMs, we use the Flickr100k for both Holidays and Oxford5k. When evaluating Oxford105k, where Flickr100k is part of the database, we us instead the *Paris* dataset [30] as negative samples.

6.2. Kernels

We have tested two different kernels, each with a scalar parameter γ.

Gaussian SLEM:

$$k_1(x, y) = e^{-\gamma\|x-y\|^2}; \tag{17}$$

Poly SLEM:

$$k_2(x, y) = x^T y + \gamma(x^T y)^2. \tag{18}$$

6.3. Base visual features

We test our feature encoder for four different base features: the hand-crafted VLAD image representation and three learned features derived from the activation coefficients of deep Convolutional Neural Networks.

We use the same VLAD variant of [10] used in [41] that relies on densely-extracted rootSIFT [2] local descriptors, per-cluster normalization, PCA-based rotations, and root normalization. Like [41], we use 64 clusters, for a final feature of size 8192.

The first CNN features we use consist of the activation coefficients of the previous-to-last layer of the AlexNet architecture [22], based on a publicly available pre-trained model [19]. These are also the features used in [41].

The SPoC features [3], which are tailored specifically for the image retrieval application, consist of spatially-weighted sums of the activations of the last convolutional layer of the 19-layer VGG network [36].

Finally, we use the NetVLAD features [1], trained for place recognition. These features are obtained by adding a differentiable version of the VLAD algorithm [10] as a layer at the end of a convolutional architecture.

6.4. Image retrieval results

We use all the those base features of the previous subsection as baseline. Since Babenko and Lemptisky [3] and Arandjelović *et al*. [1] have improved retrieval results by applying PCA followed by whitening to their features, we also apply this post-processing to our base features as a second baseline (PCAW), compressing base feature dimension to half of the original. We then compare the baselines with the original ESVM, LDA and several variants of our approach (SLEM), since all the those methods are based on similar ideas. The results are presented in Table 1. For the large-scale dataset that is Oxford105k, we limit our experiments to our best performing base features, SPoC and NetVLAD.

Linear SLEM performs similarly to ESVM while being much more time efficient (Fig. 1). The fact that a

Dataset	Holidays				Oxford 5k				Oxford 105k	
Model, features	VLAD	SPoC	AlexNet	NetVLAD	VLAD	SPoC	AlexNet	NetVLAD	SPoC	NetVLAD
Baseline	72.7	76.5	68.2	85.4	46.3	54.4	40.6	67.5	50.1	65.6
PCAW	75.5	81.7	69.2	88.3	50.9	63.7	45.0	69.1	55.5	66.1
LDA	54.7	82.2	64.1	74.3	29.6	62.2	42.5	72.7	52.4	40.7
ESVM [41]	77.5^3	84.0^3	71.3	91.4^2	57.2^3	62.1	43.9	72.5	56.5	67.5
Linear SLEM	78.0^2	82.3	72.1	91.3^3	**59.3**	64.1^3	46.2^3	72.9^3	56.7^3	68.0^3
Gaussian SLEM (16)	76.8	80.3	71.2	91.4^2	52.8	63.0	43.5	71.9	55.8	67.4
Gaussian SLEM (32)	77.4	81.7	72.0^3	91.4^2	54.9	63.1	44.0	71.1	56.0	67.8
Gaussian SLEM (fr)	**78.1**	86.2^2	**72.9**	**91.7**	59.0^2	**64.9**	47.0^2	**74.4**	59.5^2	70.0^2
Poly SLEM (16)	76.9	82.3	71.4	91.3^3	53.0	63.6	43.6	71.4	56.1	67.5
Poly SLEM (32)	77.3	82.4	72.1^2	**91.7**	54.9	63.6	44.1	71.6	56.3	67.9
Poly SLEM (fr)	**78.1**	**86.3**	**72.9**	**91.7**	**59.3**	64.8^2	**47.3**	74.1^2	**62.5**	**70.2**

Table 1: Mean average precision (mAP) results for INRIA Holidays and Oxford buildings datasets, expressed as percentages. In this table, we present our results for VLAD [10], sum-pooling of convolutional features (SPoC) [3], activation coefficients from the previous-to-last CNN layer (AlexNet) [22] and activation of NetVLAD layer [1]. In parentheses, the rank of he decomposition ('fr' for full rank decomposition). For each column, we show in **bold** the best results and index the second and third best.

hinge-loss classifier does not outperform a square-loss classifier can seem counter-intuitive, but both have been shown to be equivalent for binary classification under mild constraints [40].

We use both Gaussian SLEM and Polynomial SLEM with two decompositions: one full-rank CCD decomposition indicated by (fr) and two low-rank KPCA decompositions indicated by the rank of the decomposition. We train our exemplar classifiers for 15000 negative samples. For all the experiments we calibrate the regularization cost λ, as well as the parameter γ similarly to the calibration in [41].

The full-rank variant outperforms all methods for all base features, although the gains when compared to linear SLEM are not always significant (*e.g.* for VLAD features). We notice significant improvement for SPoC in both Holidays and Oxford and for AlexNet and NetVLAD in Oxford.

6.5. Time and storage scalability

In this section we compare the time efficiency of our method and the ESVM, as well as discuss which method and decomposition to use according to the number of negative samples.

In Fig. 1, we see that the linear SLEM efficiency does not change with n. Indeed, if d is the dimension of the base representation, A is a $d \times d$ matrix for linear SLEM, whereas for a full-rank kernel, A is $n \times n$. This explains the increasing running time for Gaussian and polynomial kernels: storage and solving a $n \times n$ system does not scale for large number of negative samples.

Retrieval results for full-rank kernelized SLEM in Fig. 1 suggest we can benefit from larger sets of negative samples. We, however, limit our full-rank experiments to $n = 15000$ negative samples due to the $O(n^3)$ complexity of the offline step. When we consider only the online procedure of our model, *i.e.* the calculation of $\beta^\star$, our kernelized model has a similar time efficiency to ESVM. Therefore, we can process the kernel SLEM for the Gaussian and polynomial kernels in similar running time to ESVM if we pre-process our negative samples offline.

For low-rank decompositions, we present in Fig. 2 a comparison in average precision between KPCA and ICD decompositions using SPoC on the Holidays dataset, fixing n and varying r. The superior results justify our preference for KPCA, despite its less efficient offline step. The only advantage of ICD over KPCA is its time complexity, linear in the number of negative samples, that allows a bigger number of negative samples. In Fig. 3 we show results for ICD for bigger pools of negative samples, to which a KPCA decomposition would be too time consuming. The results suggest that the performance of ICD SLEMs are not sensitive to the number of negative examples at a fixed small rank.

As shown by Fig. 2, the mAP for the low-rank KPCA approximation increases with the rank. Its maximum value for the features of the figure is 86.3 for full rank, $r = 15,000$ (Table 1, col. 2). Figure 2 also shows, however, that reasonable mAP values (around 84) are obtained for a much smaller rank of 200. In keeping with the usual practice in image retrieval, we limit even further the rank in the results shown in Tables 1 and 2, with very small ranks (16 and 32) that yield a total feature dimension similar to the base representation, and allow a direct comparison to methods using

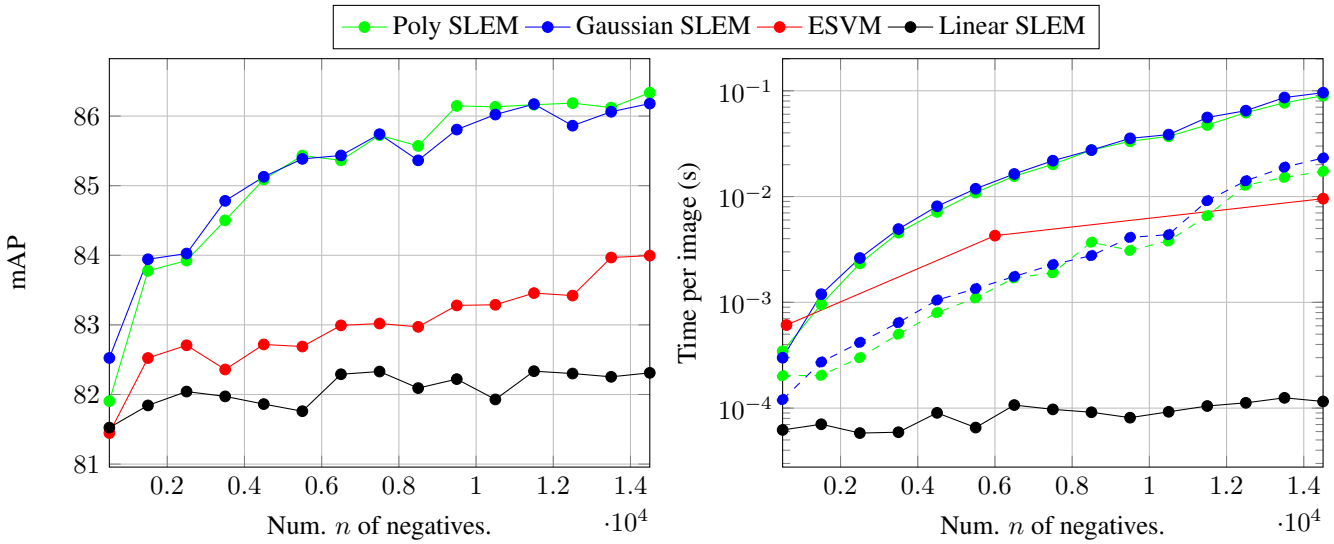

Figure 1: Results for INRIA Holidays, using SPoC features and different variants of full-rank SLEM. We use $T = 10^5$ iterations for all n to report mAP for ESVM, as suggested by [41], but report timings using $T = 1.66n$ and the values reported in Table 1 of [41]. Left: mAP; Right: computation time in solid line, *online* computational cost in dashed line.

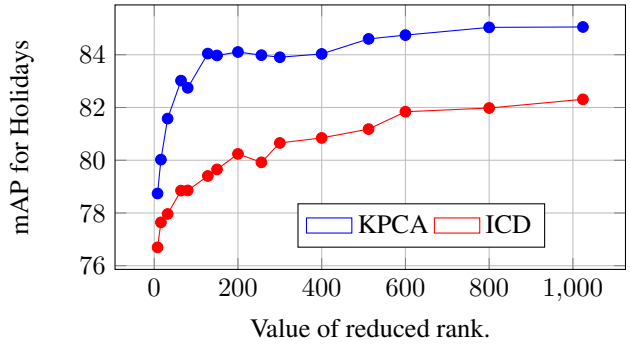

Figure 2: mAP for Holidays using SPoC + Poly SLEM for $n = 15000$ negatives. We perform two low-rank decompositions and compare its results at similar ranks.

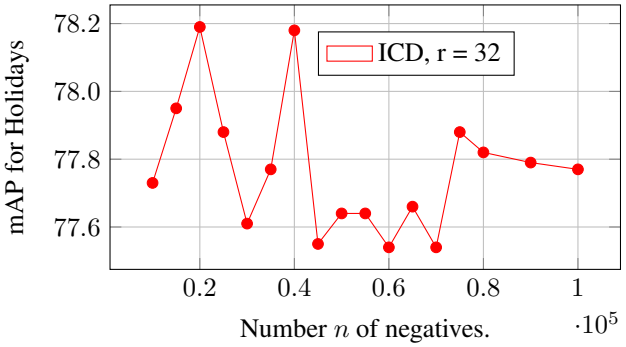

Figure 3: mAP for Holidays using SPoC + Poly SLEM, using ICD and fixed 32-rank.

these features directly. This rather extreme compression is also justified in part by the fact that these very-low rank factorizations already capture a reasonable part of the problem structure. Indeed, the relative residual error for SPoC features on Holiday with 15,000 negatives is only 0.39 for $r = 16$ and 0.31 for $r = 32$. For reference, the relative error decreases to 0.08 for $r = 600$, and 0.05 for $r = 1024$.

6.6. Comparation to the state of the art

We compare the state-of-the-art global descriptors for Holidays and Oxford 5k to both SPoC and NetVLAD features improved by linear SLEM and low-rank Poly SLEM in Table 2. We do not include re-ranking nor query expansion. We perform PCA and whitening to compress both descriptors to 256 and 512, as done in [1, 3] and compare the results by brackets of dimension. We also add a bracket of the full 4096-dimension NetVLAD for completeness, so we include our best performance. Our approach outperforms the state of the art for Holidays by 2.5 points in 256 dimensions and 0.8 in 512 dimensions, despite not using the best performing descriptors [14] as base features.

7. Conclusion and future work

In this paper, we have addressed the problem of image retrieval using the kernelized square-loss exemplar machines, and its efficient implementation. The main novelty of the paper is two-fold: First, using the square loss, which avoids retraining for each additional positive training exam-

Features	rank	dim	**Hol.**	**Ox5k**
Babenko *et al.*[3]	-	256	80.2	58.9
Radenović *et al.* [31]	-	256	81.5	<u>77.4</u>
Arandjelović *et al.* [**?**]	-	256	86.0	62.5
Kalantidis *et al.* [20]	-	256	83.1	65.4
SPoC + Linear SLEM	-	256	81.5	64.7
SPoC + Poly SLEM	16	288	80.1	63.6
SPoC + Poly SLEM	32	320	81.8	63.6
NetVLAD + Linear SLEM	-	256	<u>88.5</u>	65.9
NetVLAD + Poly SLEM	16	288	87.7	65.5
NetVLAD + Poly SLEM	32	320	88.3	65.6
Radenović *et al.* [31]	-	512	82.5	79.7
Arandjelović *et al.* [1]	-	512	86.7	65.6
Kalantidis *et al.* [20]	-	512	84.9	68.2
Gordo *et al.* [14]	-	512	89.1†	**83.1**†
SPoC + Linear SLEM	-	512	82.3	64.1
SPoC + Poly SLEM	16	544	82.3	63.0
SPoC + Poly SLEM	32	576	82.4	63.1
NetVLAD + Linear SLEM	-	512	89.3	72.3
NetVLAD + Poly SLEM	16	544	<u>89.9</u>	71.9
NetVLAD + Poly SLEM	32	576	<u>89.9</u>	72.3
Arandjelović *et al.* [1]	-	4096	88.3	69.1
NetVLAD + Linear SLEM	-	4096	91.3	72.9
NetVLAD + Poly SLEM	16	4128	91.3	71.2
NetVLAD + Poly SLEM	32	4160	**91.7**	71.7

Table 2: Compared results to state-of-the-art features at similar dimensions, without re-ranking or query augmentation. The results using Poly SLEM add 32 or 64 dimensions to the original feature (for $r = 16$ or $r = 32$, respectively). Underlined results are the best at each dimension bracket and bold results are the general best. † indicates the previous state-of-the-art.

ple and calibrating one of its parameters; second, kernelizing the method while keeping a reasonable memory footprint through the use of low-rank approximations. Similar ideas have of course been used in other contexts in machine learning [5, 11, 35, 37, 40]. Our work is, however, to our knowledge, the first to apply these ideas to examplar-based classifiers, in particular in the context of image retrieval. We have obtained significant improvements over the base features we tested and outperformed similar encoders on different datasets. As future work, we plan to work on a convolutional implementation similar to [1] so its parameters can be learned in a supervised manner. The use of other kernel functions is worth investigating. The polynomial kernel performs similarly to the Gaussian kernel, even though the Hilbert space obtained from the Gaussian kernel has infinite dimensions and the Hilbert space obtained from the polynomial kernel does not. Different kernels such as the spatial pyramid kernel [23] are another option, which would increase the versatility of our approach. Finally, our method constructs a generic feature encoding and therefore can be used in many other computer vision problems, such as object classification and scene recognition.

Acknowledgments

This work was supported by the ERC grant VideoWorld.

References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2016.

[2] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2012.

[3] A. Babenko and V. Lempitisky. Aggregating deep convolutional features for image retrieval. In *Proc. European Conf. Comp. Vision*, 2015.

[4] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitisky. Neural codes for image retrieval. In *Proc. European Conf. Comp. Vision*, 2014.

[5] F. Bach and M. Jordan. Kernel independent component analysis. In *Journal of Machine Learning Research*, 2002.

[6] F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In *Proc. Int. Conf. on Machine Learning*, 2005.

[7] C. Bilen, J. Zepeda, and P. Pérez. Learning sparsity inducing analysis operators for discriminative similarity metrics. In *Signal Processing with Adaptive Sparse Structured Representations*, 2015.

[8] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, 2010.

[9] G. C. Cawley and N. L. C. Talbot. Efficient cross-validation of kernel fisher discriminant classifiers. *Pattern Recognition*, 36(11):2585–2592, 2003.

[10] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez. Revisiting the VLAD image representation. In *Proceedings of ACM International Conference on Multimedia*, volume 21, pages 653–656, New York, New York, USA, 2013. ACM Press.

[11] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *JMLR*, 2:243–264, 2001.

[12] M. Gharbi, T. Malisiewicz, S. Paris, and F. Durand. A Gaussian approximation of feature space for fast image similarity. Technical Report CSAIL-TR-2012-032, MIT, 2012.

[13] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[14] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *Proc. European Conf. Comp. Vision*, 2016.

[15] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *Proc. European Conf. Comp. Vision*, 2012.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2009. 2nd edition.

[17] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometry consistency for large scale image search. In *Proc. European Conf. Comp. Vision*, 2008.

[18] H. Jégou and A. Zisserman. Triangulation embedding and democratic aggregation for image search. 2014.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[20] Y. Kalantidis, C. Mellina, and S. Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *Proc. European Conf. Comp. Vision*, 2016.

[21] T. Kobayashi. Three viewpoints toward exemplar-svm. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2015.

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*, pages 1–9, 2012.

[23] S. Labeznik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognizing natual scenes categories. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2006.

[24] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Proc. Int. Conf. Comp. Vision*, 2011.

[25] T. Malisiewicz, A. Shrivastava, A. Gupta, and A. A. Efros. Exemplar-svms for visual object detection, label transfer and image retrieval. In *Proc. Int. Conf. on Machine Learning*, 2012.

[26] N. Murray and F. Perronnin. Generalized max pooling. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2014.

[27] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vector. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2010.

[28] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *European Conference on Computer Vision*, 2010.

[29] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2007.

[30] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2008.

[31] F. Radenović, G. Tolias, and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *Proc. European Conf. Comp. Vision*, 2016.

[32] A. Rana, J. Zepeda, and P. Perez. Feature learning for the image retrieval task. In *Asian Computer Vision and Pattern Recognition Workshops*, 2014.

[33] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *Int. J. of Comp. Vision*, 3:222–245, 2013.

[34] B. Schölkopf, R. Herbrich, and A. Smola. A genelarized representer theorem. In *Annual Conference on Computational Learning Theory*, pages 416–426, 2001.

[35] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(3):1299–1319, 1998.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. on Learning Representations*, 2015.

[37] J. Suykens, T. V. Gestel, J. D. Moor, and J. Vandewalle. Least squares support vector machines. 2002.

[38] G. Washba. *Spline models for observational data*. SIAM, 1990.

[39] M. A. Woodbury. *Inverting modified matrices.* Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950, 1950. 4pp.

[40] J. Ye and T. Xiong. Svm versus least squares svm. In *Journal of Machine Learning Research*, 2007.

[41] J. Zepeda and P. Pérez. Examplar SVMs as visual feature encoders. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2015.