Kernel Square-Loss Exemplar Machines for Image Retrieval
Rafael S Rezende, Joaquin Zepeda, Jean S Ponce, Francis S Bach, Patrick Pérez

To cite this version:
Rafael S Rezende, Joaquin Zepeda, Jean S Ponce, Francis S 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://inria.hal.science/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. Rezende1 Joaquin Zepeda2,3 Jean Ponce1,4 Francis Bach1 Patrick Pérez2
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
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 [20] 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
multiscale representations [8] individually for each positive sample. The next section shows how to calculate all exemplar classifiers simultaneously by changing the loss function.

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
\( X = [x_1, x_2, \ldots, x_n] \) 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 \( l : \{-1, 1\} \times \mathbb{R} \rightarrow \mathbb{R}^+ \). Learning an exemplar classifier from these examples amounts to minimizing the function

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

w.r.t. \( \omega \) in \( \mathbb{R}^d \) and \( \nu \) in \( \mathbb{R} \). In Eq. (1), \( \lambda \) and \( \theta \) are respectively a regularization parameter on \( \omega \) 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 \( \omega^*(x_0, X) \) that minimizes the loss function \( J \):

\[
(\omega^*, \nu^*) = \text{argmin}_{(\omega, \nu) \in \mathbb{R}^d \times \mathbb{R}} J(\omega, \nu).
\]

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

\[
\begin{align*}
\omega^* &= \frac{2\theta}{\theta + 1} U^{-1}(x_0 - \mu), \\
\nu^* &= \frac{\theta - 1}{\theta + 1} - \frac{1}{\theta + 1} (\theta x_0 + \mu)^T \omega^*,
\end{align*}
\]

where:

\[
\begin{align*}
\mu &= \frac{1}{n} \sum_{i=1}^n x_i, \\
U &= \frac{1}{n} XX^T - \mu \mu^T + \frac{\theta}{\theta + 1} (x_0 - \mu)(x_0 - \mu)^T + \lambda I_{d_d},
\end{align*}
\]

where \( I_{d_d} \) is the identity matrix of size \( d \).

Woodbury identity. We can simplify Eq. (3) by modifying \( U \) in Eq. (4). Let us define \( A = \frac{1}{n} XX^T - \mu \mu^T + \lambda I_{d_d} \) as the regularized covariance matrix and assume its inverse \( A^{-1} \) known. The matrix \( U \) now reads

\[
U = A + \frac{\theta}{\theta + 1} \delta^T,
\]

where \( \delta = x_0 - \mu \) is the centered (w.r.t. the negatives’ mean) positive sample. The Woodbury identity [39] gives us

\[
U^{-1} = A^{-1} - \frac{\theta}{\theta \delta^T A^{-1} \delta + \theta + 1} A^{-1} \delta^T \delta A^{-1}.
\]

Substituting (5) in (3) yields

\[
\omega^* = \frac{2\theta}{\theta + 1} \left[ A^{-1} \delta - \frac{\theta}{\theta \delta^T A^{-1} \delta + \theta + 1} A^{-1} \delta^T A^{-1} \delta \right]
\]

\[
= \frac{2\theta}{\theta \delta^T A^{-1} \delta + \theta + 1} A^{-1} \delta.
\]

1Depending on the loss function \( l \), \( \nu^*(x_0, X) \) 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 $\theta$ does not influence the direction of the optimal vector $\omega^*$, only its norm. This means that if search and ranking are based on the normalized feature $\frac{1}{\|x\|} x$, e.g. using cosine similarity, $\theta$ does not influence the matching score of the SLEM vectors of two different images. This sets SLEM apart from ESVM which requires this parameter to be calibrated [24, 41]. We can thus set the value of $\theta$ 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 $\Sigma$ 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

$$
\begin{align*}
\omega &= \Sigma^{-1}(x_0 - \mu), \\
\nu &= -\frac{1}{2} (x_0 + \mu)^T \omega.
\end{align*}
$$

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 $\nu^* = \nu$, $A = \Sigma$ and that the vectors $\omega$ of Eq. (7) and $\omega^*$ 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 $\varphi$ 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 $H \times \mathbb{R}$:

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

where the pairs $(x_i, y_i)$ in $X \times \{-1, 1\}$, $i = 1 \ldots n$ are training samples, and $(h, h')$ 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 $\nu$, $(h, h') + \nu$ is an affine function of $h'$, whose zero set defines an affine hyper-plane 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 $k_i^T = [k_{i1}, k_{i2}, \ldots, k_{in}]$, $i \in \{1, \ldots, n\}$. We assume from now on that $l$ is convex and continuous. Under this assumption, Eq. (8) admits an equivalent formulation

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

and any solution $(\alpha^*, \nu^*)$ to (8) provides a solution $(h^*, \nu^*)$ to (8) with $h^* = \sum_{i=1}^{n} \alpha_i^* \varphi(x_i) + \nu^*$. 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

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

where $b_i^T$ denotes the $i$-th row of $B$ and $r$ is the number of columns of $B$. If $(\beta^*, \nu^*)$ is the solution of Eq. (10), the corresponding vector $\alpha^*$ (or, more correctly, a corresponding vector of dimension $n \geq r$) can be computed by $\alpha^* = P \beta^*$, 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 $\theta = \frac{1}{\lambda}$ (we set this value of $\theta$ 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 $A = \frac{1}{n} BB^T - \mu \mu^T + \lambda Id_r$. 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,

\[
K' = \begin{bmatrix} k_{00} & k_0^T \\ k_0 & K \end{bmatrix},
\]

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

\[
B' = \begin{bmatrix} u \\ 0 \end{bmatrix}, \quad v = B'k_0, \quad u = \sqrt{k_{00} - ||v||^2},
\]

where \( B' \) is the pseudo-inverse of \( B \).

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

\[
B'B'^T = \begin{bmatrix} u^2 + ||v||^2 \\ Bv \end{bmatrix} = \begin{bmatrix} k_{00} & v^T B \\ v & v B \\ Bv & K \end{bmatrix}.
\]

Since \( K' \) is positive semidefinite, \( k_{00} \) must lie in the column space \( B \) of \( B \). Indeed, if we suppose \( k_{00} \) does not belong to \( B \), then it can be decomposed uniquely as \( k_{00} = s + t \), \( s \in B \) and \( t \in B^\perp \), with \( t \neq 0 \). In one hand, \( K' \) being semidefinite positive implies that \( [1, -at^T]K'[1; -at] = k_{00} - 2a||t||^2 \geq 0 \) for all real value \( a \). In the other hand, for \( a \) large enough, \( k_{00} - 2a||t||^2 \leq 0 \), which is a contradiction. Hence \( v = B'k_0 \) 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 \( K - k_0k_0^T/k_{00} \) of \( k_{00} \) in \( K' \) is itself positive semidefinite. Indeed, since the matrix \( k_{00}K - k_0k_0^T = B(k_{00}Id_1 - vv^T)B^T \) is also positive semidefinite. Thus \( v^T(k_{00}Id_1 - vv^T)v = ||v||^2(k_{00} - ||v||^2) \geq 0 \).

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

\[
\frac{1}{n}(b_0^T \beta + \nu - 1)^2 + \sum_{i=1}^{n} (b_i^T \beta + \nu + 1)^2 + \frac{\lambda}{2} \| \beta \|^2,
\]

(14)

with \( b_i^T \) being the \((i + 1)\)-th row of \( B' \), \( i \in \{ 0, 1, \ldots, n \} \). In particular, \( b_0^T = [u^T; v] \) and, for \( i > 0 \), \( b_i^T = [0; b_i] \).

The solution \( (\beta^*, \nu^*) \) in \( \mathbb{R}^{r+2} \times \mathbb{R} \) can be computed just as before by Eq. (3), replacing \( x_0 \) by \( b_0^T \), \( \mu \) by \( \mu' = \frac{1}{n} \sum_{i=1}^{n} b_i^T \), and \( X \) by the \((r + 1) \times n \) matrix \( Q \) of columns \( b_1, b_2, \ldots, b_n \).

The solution \( \alpha^* \) is now calculated as \( \alpha^* = P^T \beta^* \), where

\[
P = \begin{bmatrix} u^{-1}0^T ; -u^{-1}Pv \end{bmatrix} P
\]

is the pseudoinverse of \( B'^T \).

\[
\alpha^* \text{ can be expressed by the linear system}
\]

\[
\begin{bmatrix} \alpha_0 \\
\alpha \end{bmatrix} = \begin{bmatrix} \frac{1}{2}Pv \\
-\frac{1}{2} \end{bmatrix} \beta.
\]

(15)

4.4. Similarity score

Once the optimal parameters \((\beta, \nu)\) from (14) and the coordinates \( u \) of \( b_0' \) from (12) have been found, 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 \( \nu \) and \( \nu' \) which have empirically no influence, \( s(h, h') \) is given by:

\[
s(h, h') = \langle h, h' \rangle = \alpha_0^T K \alpha + \alpha_0 k(X, x_0)^T \alpha' + \alpha_0' k(X, x_0)^T \alpha + \alpha_0 \alpha_0' k(x_0, x_0') + \alpha_0 \alpha_0' k(x_0, x_0') \nu^2(k(x_0, x_0') - v^T v).
\]

(16)

For a given image whose descriptor is \( x_0 \), we need to store \( x_0, \beta \) and \( \nu \) to calculate its similarity score to whichever other image for SLEM. Since we assume the base feature \( x_0 \) has dimension \( d \) and \( \beta \) and \( \nu \) 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 \( O(nr^2) \) 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 \( \epsilon \) to its diagonal, where \( \epsilon = \min(0, -\lambda_{\text{min}}) \) and \( \lambda_{\text{min}} \) is the smallest eigenvalue of \( K \). Therefore, \( B \) also has rank \( n \) and can be calculated by CCD from the identity \( BB^T = K + \epsilon \text{Id}_n \).

\footnote{We use matlab notation for horizontal and vertical staking.}

\footnote{We drop the “⋆” 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 \( \text{tr}(K - BB^T)/\text{tr}(K) \). From these comparisons, we set KPCA as our default low-rank decomposition. The only case ICD is more appropriate 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 use instead the Paris dataset [30] as negative samples.

6.2. Kernels

We have tested two different kernels, each with a scalar parameter \( \gamma \).

**Gaussian SLEM:**

\[ k_1(x, y) = e^{-\gamma \|x-y\|^2}; \]

**Poly SLEM:**

\[ k_2(x, y) = x^T y + \gamma (x^T y)^2. \]

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. [11] 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
suggest we can benefit from larger sets of negative samples. For large number of negative samples, the increasing running time for Gaussian and polynomial kernels: storage and solving a $n \times n$ system does not scale. 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 $\beta^*$, 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 1and 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Holidays</th>
<th></th>
<th>Oxford 5k</th>
<th></th>
<th>Oxford 105k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model, features</td>
<td>VLAD</td>
<td>SPoC</td>
<td>AlexNet</td>
<td>NetVLAD</td>
<td>VLAD</td>
</tr>
<tr>
<td>Baseline</td>
<td>72.7</td>
<td>76.5</td>
<td>68.2</td>
<td>85.4</td>
<td>46.3</td>
</tr>
<tr>
<td>PCAW</td>
<td>75.5</td>
<td>81.7</td>
<td>69.2</td>
<td>88.3</td>
<td>50.9</td>
</tr>
<tr>
<td>LDA</td>
<td>54.7</td>
<td>82.2</td>
<td>64.1</td>
<td>74.3</td>
<td>29.6</td>
</tr>
<tr>
<td>ESVM [41]</td>
<td>77.5</td>
<td>84.0</td>
<td>71.3</td>
<td>91.4</td>
<td>57.2</td>
</tr>
<tr>
<td>Linear SLEM</td>
<td>78.0</td>
<td>82.3</td>
<td>72.1</td>
<td>91.3</td>
<td>59.3</td>
</tr>
<tr>
<td>Gaussian SLEM (16)</td>
<td>76.8</td>
<td>80.3</td>
<td>71.2</td>
<td>91.4</td>
<td>52.8</td>
</tr>
<tr>
<td>Gaussian SLEM (32)</td>
<td>77.4</td>
<td>81.7</td>
<td>72.0</td>
<td>91.4</td>
<td>54.9</td>
</tr>
<tr>
<td>Gaussian SLEM (fr)</td>
<td>78.1</td>
<td>86.2</td>
<td>72.9</td>
<td>91.7</td>
<td>59.0</td>
</tr>
<tr>
<td>Poly SLEM (16)</td>
<td>76.9</td>
<td>82.3</td>
<td>71.4</td>
<td>91.3</td>
<td>53.0</td>
</tr>
<tr>
<td>Poly SLEM (32)</td>
<td>77.3</td>
<td>82.4</td>
<td>72.1</td>
<td>91.7</td>
<td>54.9</td>
</tr>
<tr>
<td>Poly SLEM (fr)</td>
<td>78.1</td>
<td>86.3</td>
<td>72.9</td>
<td>91.7</td>
<td>59.3</td>
</tr>
</tbody>
</table>

Table 1: Mean average precision (mAP) results for INRIA Holidays and Oxford buildings datasets, expressed as percentages. We notice significant improvement for SPoC in both Holidays and Oxford datasets. For all experiments we calibrate the regularization cost $\lambda$, as well as the parameter $\gamma$ 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 $\beta^*$, 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.
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. Comparison 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-
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.

<table>
<thead>
<tr>
<th>Features</th>
<th>rank</th>
<th>dim</th>
<th>Hol. Ox5k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Babenko et al. [1]</td>
<td>-</td>
<td>256</td>
<td>80.2</td>
</tr>
<tr>
<td>Radenović et al. [31]</td>
<td>-</td>
<td>256</td>
<td>81.5</td>
</tr>
<tr>
<td>Arandjelović et al. [2]</td>
<td>-</td>
<td>256</td>
<td>86.0</td>
</tr>
<tr>
<td>Kalantidis et al. [20]</td>
<td>-</td>
<td>256</td>
<td>83.1</td>
</tr>
<tr>
<td>SPoC + Linear SLEM</td>
<td>16</td>
<td>288</td>
<td>80.1</td>
</tr>
<tr>
<td>SPoC + Poly SLEM</td>
<td>32</td>
<td>320</td>
<td>81.8</td>
</tr>
<tr>
<td>NetVLAD + Linear SLEM</td>
<td>-</td>
<td>256</td>
<td>88.5</td>
</tr>
<tr>
<td>NetVLAD + Poly SLEM</td>
<td>16</td>
<td>288</td>
<td>87.7</td>
</tr>
<tr>
<td>NetVLAD + Poly SLEM</td>
<td>32</td>
<td>320</td>
<td>88.3</td>
</tr>
<tr>
<td>Radenović et al. [31]</td>
<td>-</td>
<td>512</td>
<td>82.5</td>
</tr>
<tr>
<td>Arandjelović et al. [1]</td>
<td>-</td>
<td>512</td>
<td>86.7</td>
</tr>
<tr>
<td>Kalantidis et al. [20]</td>
<td>-</td>
<td>512</td>
<td>84.9</td>
</tr>
<tr>
<td>Gordo et al. [14]</td>
<td>-</td>
<td>512</td>
<td>89.1† 83.1†</td>
</tr>
<tr>
<td>SPoC + Linear SLEM</td>
<td>16</td>
<td>544</td>
<td>82.3</td>
</tr>
<tr>
<td>SPoC + Poly SLEM</td>
<td>32</td>
<td>576</td>
<td>82.4</td>
</tr>
<tr>
<td>NetVLAD + Linear SLEM</td>
<td>-</td>
<td>512</td>
<td>89.3</td>
</tr>
<tr>
<td>NetVLAD + Poly SLEM</td>
<td>16</td>
<td>544</td>
<td>89.9</td>
</tr>
<tr>
<td>NetVLAD + Poly SLEM</td>
<td>32</td>
<td>576</td>
<td>89.2</td>
</tr>
<tr>
<td>Arandjelović et al. [1]</td>
<td>-</td>
<td>4096</td>
<td>88.3</td>
</tr>
<tr>
<td>NetVLAD + Linear SLEM</td>
<td>-</td>
<td>4096</td>
<td>91.3</td>
</tr>
<tr>
<td>NetVLAD + Poly SLEM</td>
<td>16</td>
<td>4128</td>
<td>91.3</td>
</tr>
<tr>
<td>NetVLAD + Poly SLEM</td>
<td>32</td>
<td>4160</td>
<td>91.7</td>
</tr>
</tbody>
</table>

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.

References


