Kernel Square-Loss Exemplar Machines

Anonymous CVPR submission

Paper ID 882

Abstract

Zepeda and Pérez [?] 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 a low-rank matrix decomposition and displays improved performance. Both SLEM variants exploit the fact that the negative examples are fixed for all the positives in the training set, 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 feature representations and standard image retrieval datasets.

1. Introduction

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 comparisons are done entirely based on the pixel content of the query and database images, and the search must be robust to large image variations due to camera pose, color differences and scene illumination, amongst others.

Many successful image representations for retrieval rely on unsupervised models such as K-means [?] or Gaussian mixture models [?, ?] and, before the neural networks *renaissance*, these representations would outperform methods that exploit supervised learning of image features directly [?, ?]. Today, with the success of convolutional architectures, global image descriptors are often obtained by aggregating and/or pooling the last convolutional layers [?, ?, ?] or by addition of new differentiable layers to an existing architecture [?].

The exemplar support vector machine (ESVM), originally proposed by Malisiewicz *et al.* [?], 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, a SVM classifier is learned that can generalize well, despite the drastically limited size of the set of positive examples.

Zepeda and Pérez [?] propose to treat instead the weights of the resulting classifier as a new feature vector for image retrieval. An ESVM feature is extracted from each database and query image, by treating it as the only positive sample (an *exemplar*) while keeping a fixed pool of generic negative images. Searching amounts to computing distances between the query and the 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 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., [?, ?]), and the SLEM has also been used to compare ESVMs to classical classifiers such as the linear discriminant analysis (or *LDA*) [?]. We focus here on the role of SLEMs as feature encoder for image retrieval instead of categorization.

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 that exploits 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 represen-

tation matches and even improves upon the performance of ESVM features on three standard datasets using a wide range of base features.

We also introduce a kernelized variant of SLEM that enjoys similar computational advantages and improves retrieval performances. This kernel approach allows the use of any global image feature (*e.g.* HoG [**?**], BoW [**?**], spatial pyramids [**?**]) as base representation, independent of its use on retrieval. Further computational efficiency is obtained using low-rank factorization methods, such as the incomplete Cholesky decomposition (ICD) or kernel principal components analysis (KPCA), to decompose the kernel matrix of negative samples.

The rest of this paper is organized as follows: In Section **??** we first review the original ESVM feature representation method and subsequently introduce the proposed linear SLEM model. We then introduce the kernelized SLEM variant in Section **??**. We then present the low-rank approximation of our method that enables efficient implementations in the non-linear case in Section **??**. We evaluate our proposed method in image retrieval in Section **??**, and present conclusions in Section **??**.

2. The square-loss exemplar machine

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

2.1. 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. (**??**), 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 correspondent *exemplar classifiers* of x_0 with respect to X are the weights $\omega^\star(x_0, X)$ that minimizes the loss function J:

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

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

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

2.2. 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. (**??**) 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:

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

Woodbury identity. We can simplify Eq. (**??**) by modifying U in Eq. (**??**). Let us define $A = \frac{1}{n} X X^T - \mu\mu^T + \lambda \mathrm{Id}_d$ our 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 centralized (w.r.t. the negatives' mean) positive sample. The Woodbury identity [**?**] 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 (**??**) in (**??**) yields

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

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 [**?**, **?**]. We can thus set $\theta = \frac{1}{n}$ for the remaining of this work.

2.3. LDA and SLEM

It is interesting to note the relationship between SLEM and the classical linear discriminant analysis (LDA). Let us return to Eq. (**??**) an suppose that we have multiple positive samples. It can be shown that in this case, the corresponding linear classifier of Eq. (**??**) for the square-loss is also given by (**??**), where x_0 denotes this time the center of mass of the positive samples *if* these samples 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 [**?**] for example, LDA can be seen as a (non-regularized) linear classifier with decision function $\omega_{LDA}^T z + \nu_{LDA}$, where z is a sample in $\mathbb{R}^d$, and

$$\begin{cases} \omega_{LDA} = \Sigma^{-1}(x_0 - \mu), \\ \nu_{LDA} = -\frac{1}{2}(x_0 + \mu)^T \omega_{LDA}. \end{cases} \tag{7}$$

This shows that SLEM is a generalized version of LDA: Indeed, taking $\lambda = 0$ (*i.e.* no regularization), $A = \Sigma$ and the vectors ω_{LDA} of Eq. (**??**) and $\omega^\star$ of Eq. (**??**) have the same direction, reducing SLEM to LDA. Many interesting properties of LDA have been used recently for classification tasks [**?**, **?**] and, more recently, for image retrieval with convolutional neural network activations [**?**]. With our simple generalization of LDA, we hope to obtain superior results.

3. The kernel SLEM

3.1. 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 (**??**) *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. (**??**) 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 (**??**) provides a solution $(h^\star, \nu^\star)$ to (**??**) with $h^\star = \sum_{i=1}^n \alpha_i^\star \varphi(x_i) + \nu^\star$. This result follows from the Riesz representation theorem [**?**, **?**].

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 (**??**), 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. (**??**) allows us to write the kernel learning problem (**??**) as an instance of Eq. (**??**) by setting $y_i = -1$ for all but one training sample. For our approach, we wish to solve (**??**) for many positive training samples (one at a time) 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 (**??**).

3.2. Offline preprocess of negative samples

In order to calculate offline all operations that are dependent only on negative samples, let us suppose that the positive training sample is unknown and K is the kernel matrix of the negative samples X. The preprocessing phase consists of the calculation of the decomposition B and the constants of Eq. (**??**): $\mu = \frac{1}{n} \sum_{i=1}^n b_i^T$ 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. In the processing of positive samples, *i.e.* the online phase, these results are loaded.

3.3. Online addition of a positive sample

We now wish to write Eq. (**??**) 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 the positive samples x_0. Such a matrix can be written as

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

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 show 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*

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

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

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

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

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 $[1, -at^T]K'[1; -at] = k_{00} - 2a\|t\|^2 \geq 0$ 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 $k_{00}K - k_0 k_0^T = B(k_{00}\mathrm{Id}_r - vv^T)$ 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. (**??**). With a positive exemplar, Eq. (**??**) now reads

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

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 Equation (**??**), 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 $b_1', b_2', ..., b_n'$. 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.

3.4. Similarity score

Once the optimal parameters (β, ν) from (**??**) and the coordinates u, v of b_0' from (**??**) have been found[2], they can be used directly for measuring similarity between matching images.

Indeed, suppose two image descriptors x_0 and x_0' are given and we wish to calculate the similarity score between theirs SLEM representations h and h', denoted by $s(h, h')$.

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

We write $h' = \alpha_0' \varphi(x_0') + \sum_{i=1}^n \alpha_i' \varphi(x_i) + \nu'$. Moreover, α can be expressed by the linear system

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

Using Eq. (**??**) and ignoring bias ν and ν' which have empirically no influence, $s(h, h')$ is given by:

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

For a given image whose descriptor is x_0, we need to store x_0, $\hat{\beta}$ and v to calculate its matching score to whichever other image for SLEM. Each image depends therefore on a vector of $p + 2r$ dimensions greenPP: Remind what are p and r, I lost the former!.

4. Efficient implementation

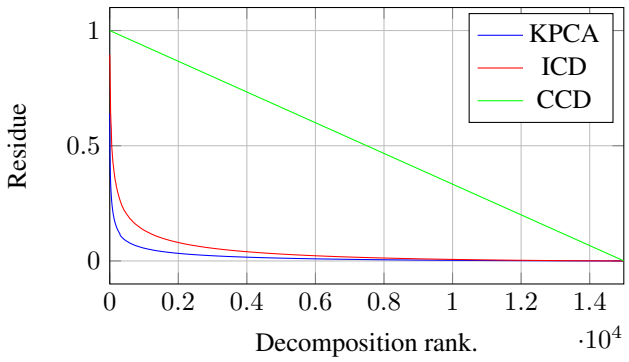
When compared to the linear square-loss classifier of Section **??**, one drawback of the kernelized approach is that the dimension of our problem grows with the size n of the negative samples. The offline factorization B of K demands $O(nr)$ storage and at best $O(nr^2)$ time. This factorization can be obtained in two fundamental ways: full-rank and low-rank decomposition. In this section we propose three different decompositions of K, to be applied depending on the compromise between accuracy and storage we went to achieve.

4.1. Full-rank decomposition

The complete Cholesky decomposition (CCD) is the most used factorization of positive-definite matrices in kernel-based learning due to its time efficiency [**?**]. 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, from the equation $BB^T = K + \epsilon \mathrm{Id}_n$, B has rank n and can be obtained by CCD.

4.2. Low-rank decomposition

Let us now turn to the problem of improving storage efficiency. One of the major requisites of a global image descriptor for large scale retrieval is to minimize the storage. As discussed in Section **??**, for each positive exemplar we store its base feature plus a $2r$ vector. Hence, we aim to decompose K at a small rank r. We consider the problem of finding a factor B of fixed rank r such that the residue $\mathrm{tr}(K - BB^T)/\mathrm{tr}(K)$ is small. yellowI wanted to



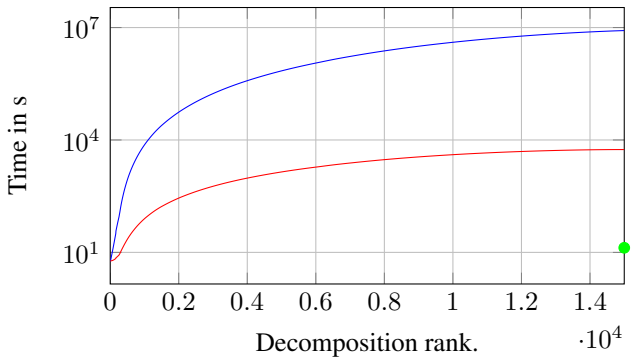


Figure 1: Comparison between complete Cholesky decomposition (CCD, in green), incomplete Cholesky decomposition (ICD, in red) and kernel PCA (KPCA, in blue), varying the rank of B. Left: Residue for each decomposition. Right: Time of calculation. We use SPoC features [**?**] of 15000 sample images.

write "factor B of fixed rank r that minimizes the residue $\mathrm{tr}(K - BB^T)/\mathrm{tr}(K)$", but I don't want to propose it as an optimization problem. The "such that the residue is small" is not perfect, but it is the best I could find. Any proposition?

The incomplete Cholesky decomposition (ICD) is widely used in machine learning [**?**, **?**, **?**]. Its algorithm is similar to CCD, and it greedily chooses which column of K to add to the decomposition based on the gain in approximation error for all non-added columns. This process is called *pivot selection* [**?**]. We stop the algorithm r steps, obtaining the factor B in time $O(nr^2)$.

The kernel PCA (KPCA) proposes that, instead of a greedy addition of columns, we choose the r first eigenvectors. This is done by doing a truncated singular value decomposition of K:

$$K = WDW^T;\ b_i = \sqrt{d_{ii}}w_i, \tag{17}$$

where w_i denotes the i-th column of W and d_{ii} the i-th diagonal element of D. Decomposing K from Equation (**??**) has complexity $O(n^2r)$ in time. [**?**]

We plot in Fig. **??** the ratio $\mathrm{tr}(K - BB^T)/\mathrm{tr}(K)$ when we vary the number of columns r of B and we see a faster convergence for the kernel PCA decomposition when r goes to n. yellowI could not finish the discussion in this section, I need some sleep...

5. Experimental Evaluation

5.1. Datasets and evaluation protocol

We perform our experiments on three standard datasets for image retrieval:

- The INRIA *Holidays* dataset [**?**], which 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 [**?**, **?**, **?**]
- The *Oxford5k* dataset [**?**], which 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 [**?**], 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 SLEM, we use the Flickr100k for both Holidays and Oxford5k. When training on Oxford105k, we us instead the *Paris* dataset [**?**] as negative samples. At evaluation time, for a dataset that consists of p images and q query images, we calculate the $p \times q$ *similarity matrix* S, where each of its q columns is the matching scores of the query image with all the p images.

5.2. Which kernel to choose?

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

$$k_1(x, y) = e^{-\gamma\|x-y\|^2};\ k_2(x, y) = x^T y + \gamma(x^T y)^2. \tag{18}$$

Gaussian SLEM. The radial basis function kernel k_1 is a well known reproducing kernel, used for classification with support vector machines.

$$k_1(x, y) = e^{-\gamma\|x-y\|^2}; \tag{19}$$

Poly SLEM. The polynomial kernel k_2 is a reproducing kernel often used in natural language processing.

$$k_2(x, y) = x^T y + \gamma(x^T y)^2. \tag{20}$$

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	-
PCAW	75.5	81.7	69.2	88.3	50.9	63.7	45.0	69.1	55.5	-
LDA	54.7	82.2	64.1	74.3	29.6	62.2	42.5	72.7	52.4	-
ESVM [?]	77.5[3]	84.0[3]	71.3	91.4[2]	57.2[3]	62.1	43.9	72.5	56.5	-
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]	-
Gaussian SLEM (16)	76.8	80.3	71.2	91.4[2]	52.8	63.0	43.5	71.9	55.8	-
Gaussian SLEM (32)	77.4	81.7	72.0[3]	91.4[2]	54.9	63.1	44.0	71.1	56.0	-
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]	-
Poly SLEM (16)	76.9	82.3	71.4	91.3[3]	53.0	63.6	43.6	71.4	56.1	-
Poly SLEM (32)	77.3	82.4	72.1[2]	**91.7**	54.9	63.6	44.1	71.6	56.3	-
Poly SLEM (fr)	**78.1**	**86.3**	**72.9**	**91.7**	**59.3**	64.8[2]	**47.3**	74.1[2]	**62.5**	-

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

5.3. 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 [?] used in [?] that relies on densely-extracted rootSIFT [?] local descriptors, per-cluster normalization, PCA-based rotations, and root normalization. Like [?], 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 [?], based on a publicly available pre-trained model [?]. These are also the features used in [?].

The SPoC features [?], which are tailored specifically for the image retrieval application, consist of spatially-weighted sums of the activations of the last convolutional layer of a 19-layer CNN [?].

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

5.4. Image retrieval results

We use the base features of the previous subsection as baseline. Since Babenko and Lemptisky [?] and Arandjelović *et al*. [?] 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 **??**. For the large-scale dataset that is Oxford105k, we limit our experiments to our best performing base features, SPoC and NetVLAD.

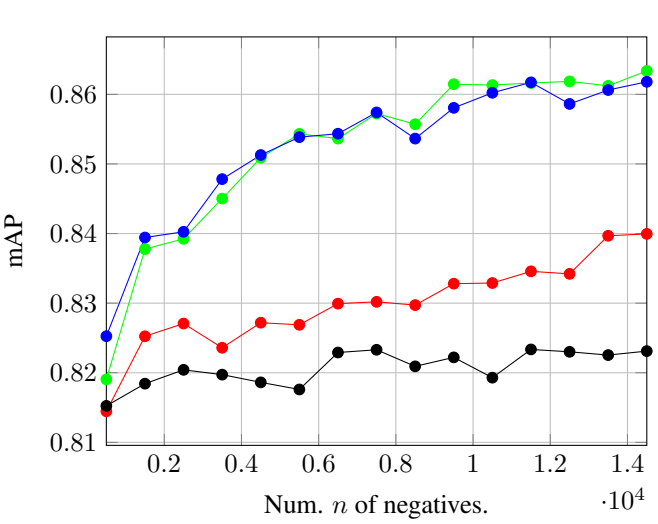
Linear SLEM performs similarly to ESVM despite being much more time efficient (Fig. **??**). The fact that a hinge-loss classifier does not outperform a square-loss classifier can seem counter-intuitive, but both have been shown to be equivalent for binary classification under mild constraints [?].

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 between 6000 and 15000 negative samplers. For all the experiments we calibrate the regularization cost λ, as well as the parameter γ similarly to the calibration in [?].

The full-rank variant outperform 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. It is interesting to notice that LDA, ESVM and Linear SLEM do not seem to outperform one another or the baselines consistently for all features and datasets.

5.5. Time and storage scalability

In this section we compare the time efficiency of our method and the E-SVM, as well as discuss which method and decomposition to use accordingly with the number of



??

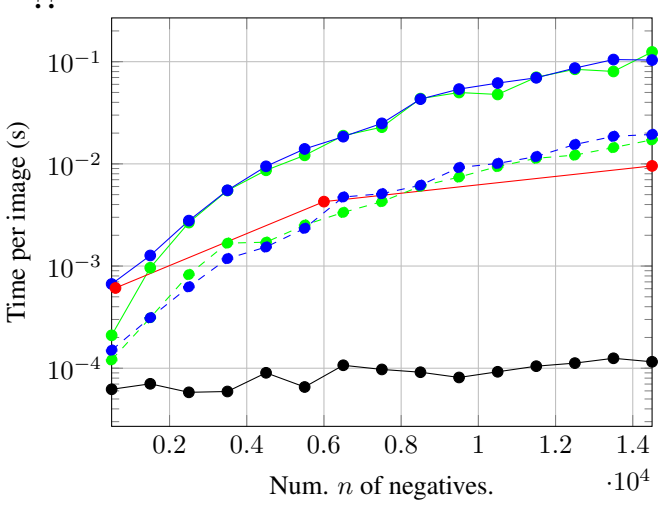


Figure 2: Results for INRIA Holidays, using SPoC features and different methods of SLEM (see legend). We use $T = 10^5$ iterations for all n to report mAP for ESVM, as suggested by [**?**], but report timings using $T = 1.66n$ and the values reported in Table 1 of [**?**]. Left: mAP; Right: computation time in solid line, *online* computational cost in dashed line.

positive and negative samples.

In Fig. **??**, we see that the see 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. **??** suggest we can benefit from larger sets of negative samples. We however limit our full-rank experiments to 15000 negatives 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 kernel similar running time to ESVM if we pre-process our negative samples offline.

For the low-rank decomposition, we present in Fig. **??** a comparison in average precision between KPCA and ICD decompositions using SPoC on the Holidays dataset. The superior results justify our preference for KPCA, despite its time-inefficient 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. **??** we show results of ICD for bigger pools of negative samples. The results suggest that performance of low-rank SLEM is not sensible to the number of negative samples.

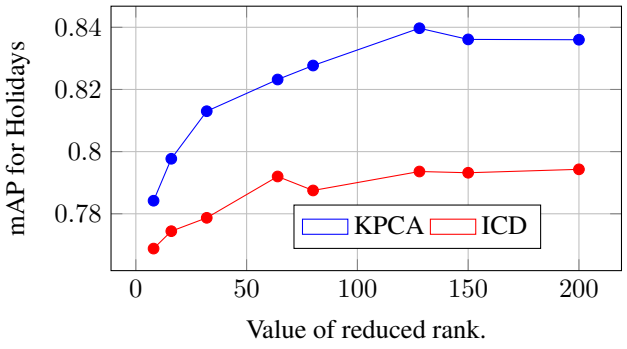


Figure 3: mAP for Holidays using SPoC + Poly SLEM. We perform two low-rank decompositions and compare its results at similar ranks.

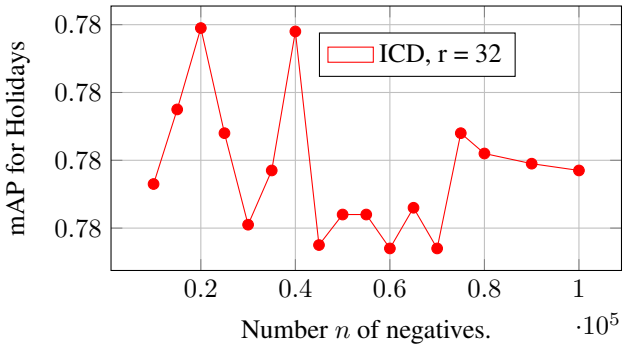


Figure 4: mAP for Holidays using SPoC + Poly SLEM, using ICD and fixed 32-rank.

Features	rank	dim	**Hol.**	**Ox5k**
Babenko *et al.*[**?**]	-	256	80.2	58.9
Radenović *et al.* [**?**]	-	256	81.5	77.4
Arandjelović *et al.* [**?**]	-	256	86.0	62.5
Kalantidis *et al.* [**?**]	-	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	88.5	65.9
NetVLAD + Poly SLEM	16	288	87.7	65.5
NetVLAD + Poly SLEM	32	320	88.3	65.6
Radenović *et al.* [**?**]	-	512	82.5	79.7
Arandjelović *et al.* [**?**]	-	512	86.7	65.6
Kalantidis *et al.* [**?**]	-	512	84.9	68.2
Gordo *et al.* [**?**]	-	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	89.9	71.9
NetVLAD + Poly SLEM	32	576	89.9	72.3
Arandjelović *et al.* [**?**]	-	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 or Gaussian 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.

5.6. Comparation to the state of the art

We compare our results to the state of the art for Holidays and Oxford 5k in Table **??**. We do not include re-ranking nor query expansion. We compare the state of the art global descriptors to both SPoC and NetVLAD features improved by Linear SLEM and low-rank Poly SLEM for rank equals to 16 and 32. We perform PCA and whitening to reduce the dimension of both descriptors to 256 and 512, and compare results by bracket 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, despite not using descriptors as base feature.

6. Conclusion and future work

In this paper, we address the problem of image retrieval with global image representation. We contribute to the domain by presenting a simple idea, the kernelized square loss exemplar machine, and its efficient implementation. As a result, we obtain significant improvements over the image representations we tested and outperform similar encoders on different datasets. As future work, we must work on a convolutional implementation similar to [**?**] 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 [**?**] are an option for baseline feature that are not global descriptors, wich would increase the versatility of our approach.