SVMs for Histogram-Based Image Classification

Olivier Chapelle, Patrick Haffner and Vladimir Vapnik

Abstract— **Traditional classification approaches generalize poorly on image classification tasks, because of the high dimensionality of the feature space. This paper shows that Support Vector Machines (SVM) can generalize well on difficult image classification problems where the only features are high dimensional histograms. Heavy-tailed RBF kernels of the form $K(\mathbf{x}, \mathbf{y}) = e^{-\rho \sum_i |x_i^a - y_i^a|^b}$ with $a \leq 1$ and $b \leq 2$ are evaluated on the classification of images extracted from the Corel Stock Photo Collection and shown to far outperform traditional polynomial or Gaussian RBF kernels. Moreover, we observed that a simple remapping of the input $x_i \rightarrow x_i^a$ improves the performance of linear SVMs to such an extend that it makes them, for this problem, a valid alternative to RBF kernels.**

keywords:
Support Vector Machines, Radial Basis Functions, Image Histogram, Image Classification, Corel.

I. Introduction

Large collections of images are becoming available to the public, from photo collections to Web pages or even video databases. To index or retrieve them is a challenge which is the focus of many research projects (for instance IBM's QBIC [1]). A large part of this research work is devoted to finding suitable representations for the images, and retrieval generally involves comparisons of images. In this paper, we choose to use color histograms as an image representation because of the reasonable performance that can be obtained in spite of their extreme simplicity[2]. Using this histogram representation, our initial goal is to perform generic object classification with a "winner takes all" approach: find the one category of object that is the most likely to be present in a given image.

From classification trees to neural networks, there are many possible choices for what classifier to use. The Support Vector Machine (SVM) approach is considered a good candidate because of its high generalization performance without the need to add a-priori knowledge, even when the dimension of the input space is very high.

Intuitively, given a set of points which belongs to either one of two classes, a linear SVM finds the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyperplane. According to [3], this hyperplane minimizes the risk of misclassifying examples of the test set.

This paper follows an experimental approach, and its organization unfolds as increasingly better results are obtained through modifications of the SVM architecture. Section II provides a brief introduction to SVMs. Section III describes the image recognition problem on Corel photo images. Section IV compares SVM and KNN-based recognition techniques which are inspired by previous work. From these results, Section V explores novel techniques, by either selecting the SVM kernel, or remapping the input, that provide high image recognition performance with low computational requirements.

II. Support Vector Machines

A. Optimal Separating Hyperplanes

We give in this section a very brief introduction to support vectors. Let $(\mathbf{x}_i, y_i)_{1 \leq i \leq N}$ be a set of training examples, each example $\mathbf{x}_i \in \mathbb{R}^d$, d being the dimension of the input space, belongs to a class labeled by $y_i \in \{-1, 1\}$. The aim is to define a hyperplane which divides the set of examples such that all the points with the same label are on the same side of the hyperplane. This amounts to finding $\mathbf{w}$ and b so that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0, \ i = 1, \ldots, N \qquad (1)$$

If there exists a hyperplane satisfying Eq. (1), the set is said to be *linearly separable*. In this case, it is always possible to rescale $\mathbf{w}$ and b so that

$$\min_{1 \leq i \leq N} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \ i = 1, \ldots, N$$

i.e. so that the closest point to the hyperplane has a distance of $1/||\mathbf{w}||$. Then, Eq. (1) becomes

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \qquad (2)$$

Among the separating hyperplanes, the one for which the distance to the closest point is maximal is called *optimal separating hyperplane* (OSH). Since the distance to the closest point is $1/||\mathbf{w}||$, finding the OSH amounts to minimizing $||\mathbf{w}||^2$ under constraints (2).

The quantity $2/||\mathbf{w}||$ is called the margin, and thus the OSH is the separating hyperplane which maximizes the margin. The margin can be seen as a measure of the generalization ability: the larger the margin, the better the generalization is expected to be [4], [5].

Since $||\mathbf{w}||^2$ is convex, minimizing it under linear constraints (2) can be achieved with Lagrange multipliers. If we denote by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ the N non negative Lagrange multipliers associated with constraints (2), our optimization problem amounts to maximizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \qquad (3)$$

with $\alpha_i \geq 0$ and under constraint $\sum_{i=1}^{N} y_i \alpha_i = 0$. This can be achieved by the use of standard quadratic programming methods [6].

The authors are with the Speech and Image Processing Services Research Laboratory, AT&T Labs-Research, 100 Schulz Drive Red Bank, NJ 07701. E-mail: {chapelle,haffner,vlad}@research.att.com.

Once the vector $\boldsymbol{\alpha^0} = (\alpha_1^0, \ldots, \alpha_N^0)$ solution of the maximization problem (3) has been found, the OSH $(\mathbf{w}_0, b_0)$ has the following expansion

$$\mathbf{w}_0 = \sum_{i=1}^{N} \alpha_i^0 y_i \mathbf{x}_i \qquad (4)$$

The *support vectors* are the points for which $\alpha_i^0 > 0$ satisfy Eq. (2) with equality.

Considering the expansion (4) of $\mathbf{w}_0$, the hyperplane decision function can thus be written as

$$f(\mathbf{x}) = sgn \left(\sum_{i=1}^{N} \alpha_i^0 y_i \mathbf{x}_i \cdot \mathbf{x} + b^0 \right) \qquad (5)$$

B. Linearly non-separable case

When the data is not linearly separable, we introduce slack variables $(\xi_1, \ldots, \xi_N)$ with $\xi_i \geq 0$ [7] such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \ i = 1, \ldots, N \qquad (6)$$

to allow the possibility of examples that violate Eq. (2). The purpose of the variables ξ_i is to allow misclassified points, which have their corresponding $\xi_i > 1$. Therefore $\sum \xi_i$ is an upper bound on the number of training errors. The generalized OSH is then regarded as the solution of the following problem : minimize

$$\frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{N} \xi_i \qquad (7)$$

subject to constraints (6) and $\xi_i \geq 0$. The first term is minimized to control the learning capacity as in the separable case; the purpose of the second term is to control the number of misclassified points. The parameter C is chosen by the user, a larger C corresponding to assigning a higher penalty to errors.

SVM training requires to fix C in Eq. (7), the penalty term for misclassifications. When dealing with images, most of the time, the dimension of the input space is large (≥ 1000) compared to the size of the training set, so that the training data is generally linearly separable. Consequently, the value of C has in this case little impact on performance.

C. Nonlinear Support Vector Machines

The input data is mapped into a high-dimensional *feature space* through some nonlinear mapping chosen a priori [8]. In this feature space, the OSH is constructed.

If we replace $\mathbf{x}$ by its mapping in the feature space $\Phi(\mathbf{x})$, Eq. (3) becomes:

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

If we have $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, then only K is needed in the training algorithm and the mapping Φ is never explicitly used. Conversely, given a symmetric positive kernel $K(\mathbf{x}, \mathbf{y})$, Mercer's theorem [3] indicates us that there exists a mapping Φ such that $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$

Once a kernel K satisfying Mercer's condition has been chosen, the training algorithm consists of minimizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad (8)$$

and the decision function becomes

$$f(\mathbf{x}) = sgn \left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \qquad (9)$$

D. Multi-Class Learning

Support Vector Machines are designed for binary classification. When dealing with several classes, as in object recognition and image classification, one needs an appropriate multi-class method. Different possibilities include:
• Modify the design of the SVM, as in [9], in order to incorporate the multi-class learning directly in the quadratic solving algorithm.
• Combine several binary classifiers: "One against one" [10] applies pairwise comparisons between classes, while "One against the others" [11] compares a given class with all the others put together.

According to a comparison study [9], the accuracies of these methods are almost the same. As a consequence, we chose the one with the lowest complexity, which is "one against the others".

In the "one against the others" algorithm, n hyperplanes are constructed, where n is the number of classes. Each hyperplane separates one class from the other classes. In this way, we get n decision functions $(f_k)_{1 \leq k \leq n}$ of the form (5). The class of a new point x is given by $\arg\max_k f_k(x)$, i.e. the class with the largest decision function.

We made the assumption that every point has a single label. Nevertheless, in image classification, an image may belong to several classes as its content is not unique. It would be possible to make multi-class learning more robust, and extend it to handle multi-label classification problems by using error correcting codes [12]. This more complex approach has not been experimented in this paper.

III. The Data and its Representation

Among the many possible features that can be extracted from an image, we restrict ourselves to ones which are global and low-level (the segmentation of the image into regions, objects or relations is not in the scope of the present paper). The simplest way to represent an image is to consider its bitmap representation. Assuming the sizes of the images in the database are fixed to $h \times w$ (h for the height and w for the width), then the input data for the SVM are vectors of size $h \times w$ for grey-level images and $3 \times h \times w$ for color images. Each component of the vector is associated to a pixel in the image. Some major drawbacks of this representation are its large size and its lack of invariance with respect to translations. For these reasons, our first

choice was the histogram representation which is described presently.

A. Color Histograms

In spite of the fact that the color histogram technique is a very simple and low level method, it has shown good results in practice [2] especially for image indexing and retrieval tasks, where feature extraction has to be as simple and as fast as possible. Spatial features are lost, meaning that spatial relations between parts of an image cannot be used. This also ensures full translation and rotation invariance.

A color is represented by a three dimensional vector corresponding to a position in a color space. This leaves us to select the color space and the quantization steps in this color space. As a color space, we chose the HSV space (Hue-Saturation-Value), which is in bijection with the RGB (Red-Green-Blue) space. The reason for the choice of HSV is that it is widely used in the literature.

HSV is attractive in theory. It is considered more suitable since it separates the color components (HS) from the luminance component (V) and is less sensitive to illumination changes. Note also that distances in the HSV space correspond to perceptual differences in color in a more consistent way than in the RGB space.

However, this does not seem to matter in practice. All the experiments reported in the paper use the HSV space. For the sake of comparison, we have selected a few experiments and used the RGB space instead of the HSV space, while keeping the other conditions identical: the impact of the choice of the color space on performance was found to be minimal compared to the impacts of the other experimental conditions (choice of the kernel, remapping of the input). An explanation for this fact is that, after quantization into bins, no information about the color space is used by the classifier.

The number of bins per color component has been fixed to 16, and the dimension of each histogram is $16^3 = 4096$. Some experiments with a smaller number of bins have been undertaken, but the best results have been reached with 16 bins. We have not tried to increase this number, because it is computationally too intensive. It is preferable to compute the histogram from the highest spatial resolution available. Subsampling the image too much results in significant losses in performance. This may be explained by the fact that by subsampling, the histogram loses its sharp peaks, as pixel colors turn into averages (aliasing).

B. Selecting classes of images in the Corel Stock Photo Collection

The Corel stock photo collection consists of a set of photographs divided into about 200 categories, each one with 100 images. For our experiments, the original 200 categories have been reduced using two different labeling approaches. In the first one, named *Corel14* , we chose to keep the categories defined by Corel. For the sake of comparison, we chose the same subset of categories as [13], which are: *air shows, bears, elephants, tigers, Arabian horses, polar bears, African specialty animals, cheetahs-leopards-jaguars, bald eagles, mountains, fields, deserts, sunrises-sunsets, night scenes.* It is important to note that we had *no influence* on the choices made in *Corel14* : the classes were selected by [13] and the examples illustrating a class are the 100 images we found in a Corel category. In [13], some images which were visually deemed inconsistent with the rest of their category were removed. In the results reported in this paper, we use all 100 images in each category and kept many obvious outliers: see for instance, in Figure 2, the "polar bear alert" sign which is considered to be an image of a polar bear. With 14 categories, this results in a database of 1400 images. Note that some Corel categories come from the same batch of photographs: a system trained to classify them may only have to classify color and exposure idiosyncracies.

In an attempt to avoid these potential problems and to move towards a more generic classification, we also defined a second labeling approach, *Corel7* , in which we designed our own seven categories: *airplanes, birds, boats, buildings, fish, people, vehicles.* The number of images in each category varies from 300 to 625 for a total of 2670 samples.

For each category images were hand-picked from several original Corel categories. For example, the *airplanes* category includes images of *air shows, aviation photography, fighter jets* and *WW-II planes.* The representation of what is an airplane is then more general. Table I shows the origin of the images for each category.

IV. SELECTING THE KERNEL

A. Introduction

The design of the SVM classifier architecture is very simple and mainly requires the choice of the kernel (the only other parameter is C). Nevertheless, it has to be chosen carefully since an inappropriate kernel can lead to poor performance. There are currently no techniques available to "learn" the form of the kernel; as a consequence, the first kernels investigated were borrowed from the pattern recognition literature. The kernel products between input vectors $\mathbf{x}$ and $\mathbf{y}$ are:

$$K_{poly}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

$$K_{Gaussian}(\mathbf{x}, \mathbf{y}) = e^{-\rho||\mathbf{x}-\mathbf{y}||_2^2}$$

K_{poly} results in a classifier which has a polynomial decision function. $K_{Gaussian}$ gives a Gaussian Radial Basis Function classifier (RBF). In the Gaussian RBF case, the number of centers (number of support vectors), the centers themselves (the support vectors), the weights (α_i) and the threshold (b) are all produced automatically by the SVM training and give excellent results compared to RBFs trained with non-SVMs methods[14].

Encouraged by the positive results obtained with $K_{Gaussian}$, we looked at generalized forms of RBF kernels:

$$K_{d-RBF}(\mathbf{x}, \mathbf{y}) = e^{-\rho d(\mathbf{x},\mathbf{y})}$$

$d(\mathbf{x}, \mathbf{y})$ can be chosen to be any distance in the input space. In the case of images as input, the L_2 norm seems to be

Category	Number of images	Corel Categories
Airplanes	386	air-shows, aviation-photography, fighter-jets, WW-II-planes
Birds	501	birds, birds2, hawks-falcons, nesting-birds, bald-eagles
Boats	200	action-sailing, sailboats,
Buildings	625	Alaska, Australia, Austrian, barns-farms, big-apple, Scotland, churches, cities-of-Italy, Egypt, Germany, pyramids, Los Angeles, Ottawa, Rome, sacred-places
Fish	300	beneath-the-Caribbean, ocean-life, underwater-life
People	358	arctic, Brazil, China-Tibet, indigenous people, people, people-of-the-world, models, rural-africa, rural-France, southeast-Asia
Vehicles	300	auto-racing, exotic-cars, military-vehicles

TABLE I

Hand-labeled categories used with the COREL database

quite meaningful and that is the reason why Gaussian RBF kernels provide good results. But as histograms are discrete densities, more suitable comparison functions exist, especially the χ^2 function, which has been used extensively for histogram comparisons [15]. We use here a symmetrized approximation of χ^2:

$$d_{\chi^2}(\mathbf{x}, \mathbf{y}) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}$$

It is not known if the kernel satisfies Mercer's condition[1].

Another obvious alternative is the L_1 distance, which gives a Laplacian RBF:

$$d_{L_1}(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$$

B. Experiments

The first series of experiments are designed to roughly assess the performance of the aforementioned input representations and SVM kernels on our two Corel tasks. The 1400 examples of *Corel14* were divided into 924 training examples and 476 test examples. The 2670 examples of *Corel7* were split evenly between 1375 training and test examples. The SVM error penalty parameter C was set to 100, which can be considered in most cases as "large". However, in this series of experiments, this parameter setting was found to enforce full separability for all types of kernels except the linear one. In the cases of the RBF kernels, the ρ values were selected heuristically. More rigorous procedures will be described in the second series of experiments.

Table II shows very similar results for both the RBG and HSV histogram representations, and also, with HSV histograms, similar behaviors between *Corel14* and *Corel7*. The "leap" in performance does not happen, as normally expected by using RBF kernels but with the proper choice of metric within the RBF kernel. Laplacian or χ^2 RBF

kernels reduce the Gaussian RBF error rate from around 30% down to 15-20%..

This improved performance is not only due to the choice of the appropriate metric, but also to the good generalization of SVMs. To demonstrate this, we conducted some experiments of image histogram classification with a K-Nearest Neighbors (KNN) algorithm with the distances χ^2 and L_2. $K = 1$ gave the best results. Table III presents the results. As expected, the χ^2 distance is better suited; the χ^2-based SVM is still roughly twice as good as the χ^2-based KNN.

	KNN L_2	KNN χ^2
Corel14	47.7	26.5
Corel7	51.4	35.4

TABLE III

Error rates with KNN

We also did some experiments using the pixel image as input to SVM classifiers with 96×64 images. Except in the linear case, the convergence of the support vector search process was problematic, often finding a hyperplane where every sample is a support vector. The error rate never dropped below 45%.

The same database has been used by [13] with a decision tree classifier and the error rate was about 50%, similar to the 47.7% error rate obtained with the traditional combination of an HSV histogram and a KNN classifier. The 14.7% error rate obtained with the Laplacian or χ^2 RBF represents a nearly four-fold reduction.

One partial explanation for the superior performance of χ^2 or Laplacian RBF kernels comes from the specific nature of the histogram representation. Let us start with an example: in many images, the largest coordinate in the histogram vector corresponds to the blue of the sky. A small shift in the color of the sky, which does not affect the nature of the object to be recognized (for instance plane or bird) results into a large L_2 distance.

Suppose a P-pixel bin in the histogram accounts for a single uniform color region in the image (with histogram

---

[1]It is still possible apply the SVM training procedure to kernels that do not satisfy Mercer's condition. What is no longer guaranteed is that the optimal hyperplane maximizes some margin in a hidden space.

Database	Input	linear	Poly 2	RBF		
				Gaussian	Laplacian	χ^2
Corel14	RGB	42.1	33.6	28.8	14.7	14.7
Corel14	HSV	36.3	35.3	30.5	14.5	14.7
Corel7	HSV	42.7	38.9	32.2	20.5	21.6

TABLE II

Error rates using the following kernels: linear, polynomial of degree 2, Gaussian RBF, Laplacian RBF and χ^2 RBF.

$\mathbf{x}$). A small change of color in this region can move the P pixels to a neighboring bin, resulting in a slightly different histogram $\mathbf{x}'$. If we assume that this neighboring bin was empty in the original histogram $\mathbf{x}$, the kernel values are:

$$K_{Gaussian}(\mathbf{x}, \mathbf{x}') = e^{-2\rho(P-0)^2} = e^{-2\rho P^2}$$

$$K_{Laplacian}(\mathbf{x}, \mathbf{x}') = e^{-2\rho|P-0|} = e^{-2\rho P}$$

$$K_{\chi^2-RBF}(\mathbf{x}, \mathbf{x}') = e^{-2\rho \frac{(P-0)^2}{P+0}} = e^{-2\rho P}$$

The kernel has a linear exponential decay in the Laplacian and χ^2 cases, while it has a quadratic exponential decay in the Gaussian case.

V. Kernel Design vs. Input Remapping

The experiments performed in the previous section show that non-Gaussian RBF kernels with exponential decay rates that are less than quadratic can lead to remarkable SVM classification performances on image histograms. This section explores two ways to reduce the decay rate of RBF kernels. It shows that one of them amounts to a simple remapping of the input, in which case the use of the kernel trick is not always necessary.

A. Non-Gaussian RBF kernels

We introduce kernels of the form $K(\mathbf{x}, \mathbf{y}) = e^{-\rho d_{a,b}(\mathbf{x},\mathbf{y})}$ with

$$d_{a,b}(\mathbf{x}, \mathbf{y}) = \sum_i |x_i^a - y_i^a|^b$$

The decay rate around zero is given by $d_{a,b}(\mathbf{x}, 0) = \sum_i |x_i|^{ab}$. In the case of Gaussian RBF kernels, $ab = 2$: decreasing the value of ab would provide for a slower decay.

A data-generating interpretation of RBFs is that they correspond to a mixture of local densities (generally Gaussian): in this case, lowering the value of b amounts to using heavy-tailed distributions. Such distributions have been observed in speech recognition and improved performances have been obtained by moving from $b = 2$ (Gaussian) to $b = 1$ (Laplacian) or even $b = 0.5$ (Sublinear) [16]. Note that if we assume that histograms are often distributed around 0 (only a few bins have non-zero values), decreasing the value of a should have roughly the same impact as lowering b. [2]

[2] An even more general type of Kernel is $K(\mathbf{x}, \mathbf{y}) = e^{-\rho d_{a,b,c}(\mathbf{x},\mathbf{y})}$

The choice of a has no impact on Mercer's condition as it amounts to a change of input variables.

$e^{-\rho(|\mathbf{x}-\mathbf{y}|)^b}$ satisfies Mercer's condition if and only if $0 \leq b \leq 2$ ([4] page 434).

B. Nonlinear Remapping of the Input

The exponentiation of each component of the input vector by a does not have to be interpreted in terms of kernel products. One can see it as the simplest possible nonlinear remapping of the input that does not affect the dimension.

The following gives us reasons to believe that a-exponentiation may improve robustness with respect to changes in scale. Imagine that the histogram component x_{col} is caused by the presence of color *col* in some object. If we increase the size of the object by some scaling factor s, the number of pixels is multiplied by s^2, and x_{col} is multiplied by the same factor. The a-exponentiation could lower this quadratic scaling effect to a more reasonable s^{2a}, with $a < 1$.

An interesting case is $a = 0$, which transforms all the components which are not 0 to 1 (we assume that $0^0 = 1$).

C. Experimental Setup

To avoid a combinatorial explosion of kernel/remapping combinations, it is important to restrict the number of kernels we try. We chose three types of RBF kernels: Gaussian($b = 2$), Laplacian ($b = 1$) and Sublinear ($b = 0.5$). As a basis for comparison, we also kept the linear SVMs.

For the reasons stated in Section III-A, the only image representation we consider here is the 16x16x16 HSV histogram.

Our second series of experiments attempts to define a rigorous procedure to choose C and ρ. Because we are only testing linear and RBF kernels, we can reduce these two choices to one, a multiplicative renormalization of the input data.

In the case of RBF kernels, we observed experimentally that full separability was always desirable on both *Corel7* and *Corel14* . As a consequence, C has to be chosen large enough (C=300) compared to the diameter of

with

$$d_{a,b,c}(\mathbf{x}, \mathbf{y}) = (\sum_i |x_i^a - y_i^a|^b)^c$$

Decreasing the value of c does not improve performance as much as decreasing a and b, and significantly increases the number of support vectors.

the sphere containing the input data (The distance between x and y is equal to $K(x,x) - 2K(x,y) + K(y,y) = 2 - 2K(x,y)$, which is always smaller than 2.0). However, RBF kernels still do not specify what value to choose for ρ. With proper renormalization of the input data, we can set $\rho = 1$ as

$$e^{-\rho \sum_i |x_i^a - y_i^a|^b} = e^{-\sum_i |(\lambda x_i)^a - (\lambda y_i)^a|^b} \text{ with } \lambda = \rho^{\frac{1}{ab}}$$

In the linear case, the diameter of the data depends on the way it is normalized. The choice of C is equivalent to the choice of a multiplicative factor $\lambda = \frac{1}{\sqrt{C}}$ for the input data. If, in Eq. (6), we replace $\mathbf{x}_i$ with $\lambda \mathbf{x}_i$ and $\mathbf{w}$ with $\mathbf{w}' = \frac{\mathbf{w}}{\lambda}$, Eq. (7) becomes:

$$\frac{1}{2}\mathbf{w}' \cdot \mathbf{w}' + \sum_{i=1}^N \xi_i \qquad (10)$$

Similar experimental conditions are applied to both *Corel7* and *Corel14* . Each category is divided into three sets, each containing one third of the images, used as training, validation and test sets. For each value of the input renormalization, support vectors are obtained from the training set and tested on the validation set. The λ renormalization for which we obtain the best result is then used to obtain a set of support vectors from both, the training and the validation sets. Each Corel image contains $M = 372992$ usable pixels: the 4096 histogram vector components range from 0 to M and sum up to M. They were renormalized with $\lambda \in \{10^{-4}, 10^{-2}, 1, 10^2\}$. Usually, the optimal values are 10^{-2} or 1. Non-optimal λ values increase the error rate by values ranging from 0.5% to 5%. This very sparse sampling rate was found to be sufficient for all kernels except Gaussian RBFs. In the latter case, we chose $\lambda \in \{10^{-4}, 0.0025, 0.01, 0.04, 0.16, 1, 10^2\}$.

The final performance result is measured on the test set. To obtain more test samples, we applied this procedure three times, each time with a different test set: the number of testing samples to the total number of data (1400 for *Corel14* and 2670 for *Corel7*). On *Corel14* , each of the three training sessions used 933 examples and required between 52 and 528 support vectors to separate one class from the others. On *Corel7* , each of the three training sessions used 1780 examples and required between 254 and 1008 support vectors to separate one class from the others. The algorithms and the software used to train the SVMs were designed by Osuna [17], [18].

D. Computation requirements

We also measured the computation required to classify one image histogram. Since the number of cycles required to perform an operation depends on the machine, we count the three main types of operations we find in our SVM classifiers:

`flt` basic floating point operation such as the multiply-add or the computation of the absolute value of the difference between two vector components. This is the central operation of the kernel dot product. This operation can be avoided if both components are zero, but we assume that verifying this condition usually takes more time than the operation itself. The computation of $(x_i - y_i)^2$ can be reduced to a multiply-add as x_i^2 and y_i^2 can be computed in advance.

`sqrt` square root

`exp` exponential

Except in the sublinear RBF case, the number of `flt` is the dominating factor. In the linear case, the decision function (Equation 5) allows the support vectors to be linearly combined: there is only one `flt` per class and component. In the RBF case, there is one `flt` per class, component and support vector. Because of the normalization by 7×4096, the number that appears on the table equals the number of support vectors. Fluctuations of this number are mostly caused by changes in the input normalization λ.

In the sublinear RBF case, the number of `sqrt` is dominating. `sqrt` is in theory required for each component of the kernel product: this is the number we report. It is a pessimistic upper bound since computations can be avoided for components with value 0.

E. Observations

The analysis of the Tables IV,V and VI shows the following characteristics that apply consistently to both *Corel14* and *Corel7* :

• As anticipated, decreasing a has roughly the same impact as decreasing b. (compare column $b = 2$ to line $a = 1$, on both Tables IV and V).

• For both, *Corel14* and *Corel7* , the best performance is achieved with $b = 1$ and $a = 0.25$.

• For histogram classification, Gaussian RBF kernels are hardly better than linear SVMs and require around NSV (number of support vectors) times more computations at recognition time.

• Sublinear RBF kernels are no better than Laplacian RBF kernels (provided that $a < 2$) and are too computationally intensive: a time-consuming square root is required for non-zero components of every support vector.

• For the practical use of RBF kernels, memory requirements may also be an issue. A full floating point representation of 5000 support vectors, each with 4096 components, requires 80 Megabytes of memory.

• Reducing a to 0.25 makes linear SVMs a very attractive solution for many applications: its error rate is only 30% higher than the best RBF-based SVM, while its computational and memory requirements are several orders of magnitude smaller than for the most efficient RBF-based SVM.

• Experiments with $a = 0$ yield surprisingly good results, and show that what is important about a histogram bin is not its value, but whether it contains any pixel at all. Note that in this case, Gaussian, Laplacian and sublinear RBFs are exactly equivalent.

• The input space has 4096 dimensions: this is high enough to enforce full separability in the linear case. However, when optimizing for λ with the validation set, a solution with training misclassifications was preferred (around 1%

Kernel	linear	RBF		
		Gaussian b=2	**Laplacian** b=1	**Sublinear** b=0.5
N_{SV}	2023	2307	3339	3352
a=1	36.4	32.7	17.4	13.7
a=0.5	22.2	17.1	12.7	12.6
a=0.25	15.2	13.0	11.0	12.4
a=0.125	14.3	12.0	11.5	12.5
a=0.0	18.4	16.5		

TABLE IV

Average error rates on Corel14 . Each columns corresponds to a different kernel. The first line reports the average number of support vectors required for the full recognizer (i.e. 14 "one against the others" SVM classifiers). The next lines report the error rates using nonlinear input remappings (exponentiation by a).

Kernel	linear	RBF		
		Gaussian b=2	**Laplacian** b=1	**Sublinear** b=0.5
N_{SV}	3567	4137	5127	5418
a=1	42.5	40.4	22.8	19.2
a=0.5	28.4	21.2	17.4	18.5
a=0.25	23.6	17.6	16.3	18.8
a=0.125	26.9	28.6	19.0	19.3
a=0.0	33.2	24.2		

TABLE V

Average error rates on Corel7 .

Kernel OP	linear			Gaussian			Laplacian			Sublinear		
	flt	sqrt	exp	flt	sqrt	exp	flt	sqrt	exp	flt	sqrt	exp
a=1	1	0.00	0	543	0.00	0.13	897	0.00	0.22	802	802.00	0.20
a=0.5	1	0.14	0	677	0.14	0.17	675	0.14	0.16	774	774.14	0.19
a=0.25	1	0.29	0	491	0.29	0.12	719	0.29	0.18	764	764.29	0.19
a=0.125	1	0.43	0	651	0.43	0.16	637	0.43	0.16	755	755.43	0.18

TABLE VI

Computational requirements for Corel7 , reported as the number of operations for the recognition of one example, divided by 7×4096.

error on the case of *Corel14* and 5% error in the case of *Corel7*).

Table VII presents the class-confusion matrix corresponding to the use of the Laplacian kernel on *Corel7* with $a = 0.25$ and $b = 1.0$ (these values yield the best results for both *Corel7* and *Corel14*). The most common confusions happen between *birds* and *airplanes*, which is consistent.

VI. SUMMARY

In this paper, we have shown that it is possible to push the classification performance obtained on image histograms to surprisingly high levels with error rates as low as 11% for the classification of 14 Corel categories and 16% for a more generic set of objects. This is achieved without any other knowledge about the task than the fact that the input is some sort of color histogram or discrete density.

This extremely good performance is due to the superior generalization ability of Support Vector Machines (SVMs) in high dimensional spaces to the use of heavy-tailed Radial Basis Function (RBFs) as kernels and to nonlinear transformations applied to the histogram bin values. We studied how the choice of the L_p distance used in a RBF kernel affects performance on histogram classification, and found Laplacian RBF kernels to be superior to the standard Gaussian RBF kernels. As a nonlinear transformation of the bin values, we used a-exponentiation with a ranging from 1 down to 0. In the case of RBF kernels, the lowering of a and p have similar effects, and their combined influence yields the best performance.

The lowering of a improves the performance of linear

	Airpl.	Birds	Boats	Build.	Fish	People	Vehic.
Airplanes	341	22	7	4	0	0	12
Birds	30	402	7	27	17	11	7
Boats	8	8	163	11	3	4	3
Buildings	6	29	10	535	11	19	15
Fish	2	16	3	10	253	13	3
People	3	13	0	29	15	296	2
Vehicles	12	12	3	22	5	2	244

TABLE VII

Class-confusion matrix for $a = 0.25$ and $b = 1.0$. For example, row (1) indicates that on the 386 images of the Airplanes category, 341 have been correctly classified, 22 have been classified in Birds, 7 in Boats, 4 in Buildings and 12 in Vehicles.

SVMs to such an extent that it makes them a valid alternative to RBF kernels, giving comparable performance for a fraction of the computational and memory requirements. This suggests a new strategy for the use of SVMs when the dimension of the input space is extremely high. Rather than introducing kernels intended at making this dimension even higher, which may not be useful, it is recommended to first try nonlinear transformations of the input components in combination with linear SVMs. The computations may be orders of magnitude faster and the performances comparable.

This work can be extended in several ways. Higher-level spatial features can be added to the histogram features. Allowing for the detection of multiple objects in a single image would make this classification-based technique usable for image retrieval: an image would be described by the list of objects it contains. Histograms are used to characterize other types of data than images, and can be used, for instance, for fraud detection applications. It would be interesting to investigate if the same type of kernel brings the same gains in performance.

REFERENCES

[1] W. Niblack, R. Barber, W. Equitz, M. Flickner, D. Glasman, D. Petkovic, and P. Yanker, "The qbic project: Querying image by content using color, texture, and shape," *SPIE*, vol. 1908, pp. 173–187, February 1993.
[2] M. Swain and D. Ballard, "Indexing via color histograms," *Intern. Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.
[3] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
[4] V. Vapnik, *Statistical Learning Theory*, John Wiley, New York, 1998.
[5] P. Bartlett and J. Shawe-Taylor, "Generalization performance of support vector machines and other pattern classifiers," in *Advances in Kernel Methods - Support Vector Learning*. 1998, MIT Press - Cambridge, USA.
[6] M. Bazaraa and C.M. Shetty, *Nonlinear programming*, John Wiley, New York, 1979.
[7] C. Cortes and V. Vapnik, "Support vector network," *Machine learning*, vol. 20, pp. 1–25, 1995.
[8] B.E. Boser, I.M. Guyon, and V.N. Vapnik, "A training algorithm for optimal margin classifier," in *Proc. 5th ACM Workshop on Computational Learning Theory*, Pittsburgh, PA, July 1992, pp. 144–152.
[9] J. Weston and C. Watkins, "Multi-class support vector machines," Tech. Rep. CSD-TR-98-04, Royal Holloway, University of London, 1998.
[10] M. Pontil and A. Verri, "Support vector machines for 3-d object recognition," in *Pattern Analysis and Machine Intelligence*, june 1998, vol. 20.
[11] V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter, "Comparison of view-based object recognition algorithms using realistic 3d models," in *Artificial Neural Networks - ICANN'96*, Berlin, 1996, pp. 251–256.
[12] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Proc. of the 1998 Workshop on Computational Learning Theory*, 1998.
[13] C. Carson, S. Belongie, H. Greenspan, and J. Mlik, "Color- and texture-based images segmentation using em and its application to image querying and classification," Submitted to PAMI, 1998.
[14] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," A.I. Memo No. 1599, MIT, 1996.
[15] B. Schiele and J.L. Crowley, "Object recognition using multidimensional receptive field histograms," in *ECCV'96, Fourth European Conference on Computer Vision, Volume I*, 1996, pp. 610–619.
[16] S. Basu and C. A. Micchelli, "Parametric density estimation for the classification of acoustic feature vectors in speech recognition," in *Nonlinear Modeling: Advanced Black-Box Techniques*, J.A.K. Suykens and J. Vandewalle, Eds. Kluwer Academic Publishers, Boston, 1998.
[17] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines:an application to face detection," in *IEEE CVPR'97*, Puerto Rico, June 17-19 1997.
[18] E. Osuna, R. Freund, and F. Girosi, "Improved training algorithm for support vector machines," in *IEEE NNSP'97*, Amelia Island, FL, September 24-26 1997.
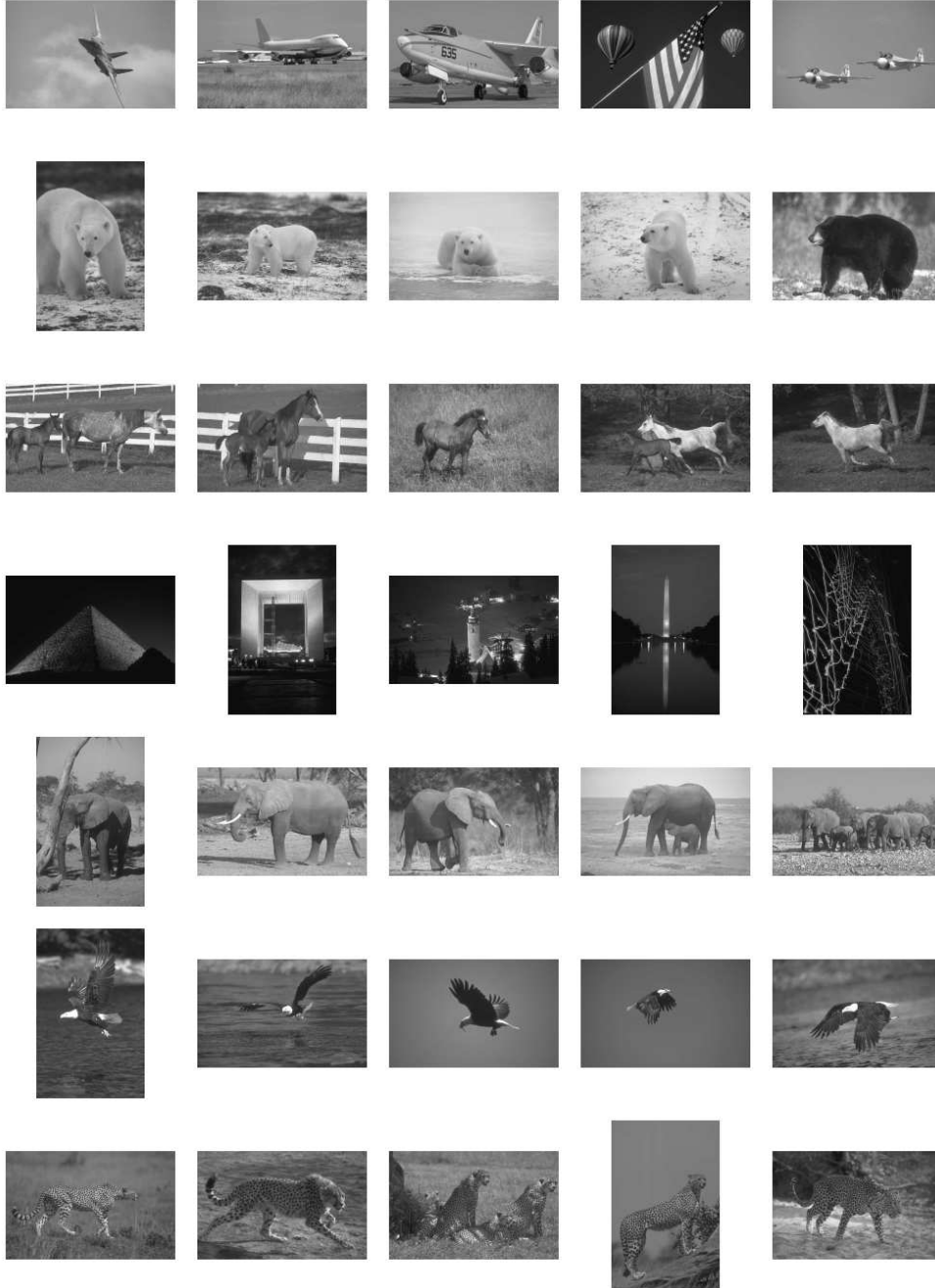
APPENDIX

I. COREL DATABASE

Fig. 1. *Corel14: each row includes images from the following 7 categories : Air Shows, Bears, Arabian Horses, Night Scenes, Elephants, Bald Eagles, Cheetahs-Leopards-Jaguars*

Fig. 2. *Corel14: each row includes images from the following 7 categories : Tigers, African Specialty Animals, Mountains, Fields, Deserts, Sunrises-Sunsets, Polar Bears.*

Fig. 3. *Corel7: each row includes images from the following categories : Airplanes, Birds, Boats, Buildings, Fish, People, Cars*