Efficient Classification for Additive Kernel SVMs

Subhransu Maji, *Member*, *IEEE*, Alexander C. Berg, *Member*, *IEEE*, and Jitendra Malik, *Fellow*, *IEEE*

Abstract—We show that a class of nonlinear kernel SVMs admits approximate classifiers with runtime and memory complexity that is independent of the number of support vectors. This class of kernels, which we refer to as *additive* kernels, includes widely used kernels for histogram-based image comparison like intersection and chi-squared kernels. Additive kernel SVMs can offer significant improvements in accuracy over linear SVMs on a wide variety of tasks while having the same runtime, making them practical for large-scale recognition or real-time detection tasks. We present experiments on a variety of datasets, including the INRIA person, Daimler-Chrysler pedestrians, UIUC Cars, Caltech-101, MNIST, and USPS digits, to demonstrate the effectiveness of our method for efficient evaluation of SVMs with additive kernels. Since its introduction, our method has become integral to various state-of-the-art systems for PASCAL VOC object detection/image classification, ImageNet Challenge, TRECVID, etc. The techniques we propose can also be applied to settings where evaluation of weighted additive kernels is required, which include kernelized versions of PCA, LDA, regression, k-means, as well as speeding up the inner loop of SVM classifier training algorithms.

Index Terms—Image classification, support vector machines, efficient classifiers, additive kernels

1 Introduction

Consider sliding window detection, one of the leading approaches for detecting objects in images like faces [42], [61], pedestrians [42], [10], [16], and cars [44]. In this approach, first, a classifier is trained to recognize an object at a fixed "pose"—for example, as shown in Fig. 1, one may train a classifier to classify 64×96 pixel pedestrians which are all centered and scaled to the same size, from background. In order to detect pedestrians at arbitrary location and scale in an image, the classifier is evaluated by varying the location and scale of the classification window. Finally, detections are obtained by finding peaks of the classification score over scales and locations, a step commonly referred to as nonmaximum suppression. Although this approach is simple—the classifier does not have to deal with invariance—a key drawback of this approach is computational complexity. On typical images these classifiers can be evaluated several tens of thousands of times. One may also want to search over aspect ratios, viewpoints, etc., compounding the problem. Therefore, efficient classifiers are crucial for effective detectors.

Discriminative classifiers based on Support Vector Machines (SVMs) and variants of boosted decision trees are two of the leading techniques used in vision tasks ranging from object detection [42], [61], [10], [16], multicategory object recognition in Caltech-101 [20], [32], to texture discrimination [67]. Classifiers based on boosted decision trees such as [61], have faster classification speed, but are significantly slower to train. Furthermore, the complexity of training can grow exponentially with the number of classes [55]. On the other hand, given the right feature space, SVMs can be more efficient during training. Part of the appeal of SVMs is that nonlinear decision boundaries can be learned using the "kernel trick" [50]. However, the runtime complexity of a nonlinear SVM classifier can be significantly higher than a linear SVM. Thus, linear kernel SVMs have become popular for real-time applications as they enjoy both faster training and faster classification, with significantly less memory requirements than nonlinear kernels.

Although linear SVMs are popular for efficiency reasons, several nonlinear kernels are used in computer vision as they provide better accuracy. Some of the most popular ones are based on comparing histograms of low-level features like color and texture computed over the image and using a kernel derived from histogram intersection or chi-squared distance to train a SVM classifier. In order to evaluate the classification function, a test histogram is compared to a histogram for each of the support vectors. The number of support vectors can often be a significant fraction of the training data, so this step is computationally very expensive as the test time scales linearly with the number of support vectors. This paper presents and analyzes a technique to greatly speed up that process for histogram comparison functions that are *additive*—where the comparison is a linear combination of functions of each coordinate of the histogram. **In particular we show it is possible to evaluate the classifier to arbitrary precision in time independent of the number of support vectors—similar to that of a linear SVM.**

This more efficient approach makes SVMs with additive kernels—used in many of the current most successful object detection/recognition algorithms—efficient enough to apply much more broadly, even possibly to real-time applications. The class of kernels includes the pyramid matching or *intersection kernels* used in Grauman and Darell [20]; and Lazebnik et al. [32]; and the chi-squared kernel used by Varma and Ray [57]; and Chum and Zisserman [8], which

- *S. Maji is with the Toyota Technological Institute at Chicago, 6045 S. Kenwood Avenue, Chicago, IL 60637. E-mail: smaji@ttic.edu.*
- *A.C. Berg is with the Computer Science Department, Stony Brook University, 1418 Computer Science, Stony Brook, NY 11794. E-mail: aberg@cs.stonybrook.edu.*
- *J. Malik is with the University of California at Berkeley, 722 Sutardja Dai Hall, Berkeley, CA 94720-1776. E-mail: malik@eecs.berkeley.edu.*

Manuscript received 31 Jan. 2011; revised 15 Dec. 2011; accepted 16 Feb. 2012; published online 28 Feb. 2012.
Recommended for acceptance by F. Bach.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-01-0070.
Digital Object Identifier no. 10.1109/TPAMI.2012.62.

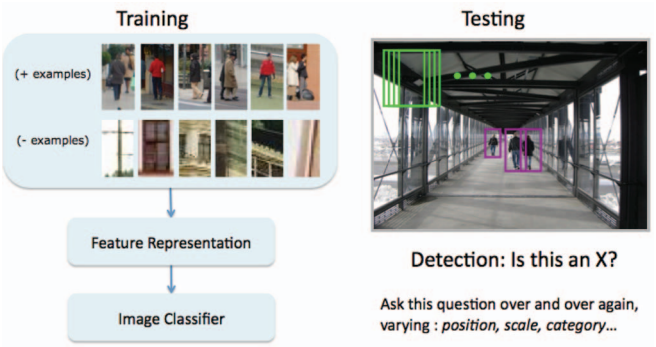

Fig. 1. A typical "sliding window" detection pipeline.

together represent some of the best results in image and object recognition on the Caltech [14] and PASCAL VOC [12] datasets.

Although the results in this paper apply to any additive kernel, we begin by analyzing the *histogram intersection* kernel, $K_{\min}(h_a, h_b) = \sum_i \min(h_a(i), h_b(i))$, that is often used as a measurement of similarity between histograms h_a and h_b. Because it is positive definite [54] for nonnegative features and conditionally positive definite for arbitrary features [34], it can be used as a kernel for discriminative classification using SVMs. Recently, intersection kernel SVMs (henceforth referred to as IKSVMs) have become popular with the introduction of pyramid match kernel [20] and spatial pyramid match kernel [32] for object detection and image classification. Unfortunately, this success typically comes at great computational expense compared to simpler linear SVMs because nonlinear kernels require memory and computation linearly proportional to the number of support vectors for classification.

In this paper, we show the following:

- SVMs using the histogram intersection kernel can be *exactly* evaluated exponentially faster than the straightforward implementation used in the previous state of the art, as has been previously shown in [25] and independently in our own work in [35] (Section 3).
- A generalization allows *arbitrary* additive kernel SVMs to be evaluated with the same "big O" computational cost as linear SVMs (Section 4), as well as significantly reducing the memory overhead, making them practical for detection and real-time applications.
- We show that *additive* kernels arise naturally in many computer vision applications (Section 5) and are already being used in many state-of-the-art recognition systems.
- Additive kernels such as histogram intersection are sufficiently general, i.e., the corresponding kernel SVM classifier can represent arbitrary additive classifiers. The difference between additive kernels can be analyzed mainly in terms of the implied regularization for a particular kernel. This helps us to understand both the potential benefit and the inherent limitations of *any* additive classifier, in addition to shedding some light on the tradeoffs between choices of additive kernels for SVMs (Section 6).
- Our approach can be computationally more efficient compared to some of the recently proposed methods and the previous state of the art in kernel classifier evaluation (Section 7).
- Combining these efficient additive classifiers with a novel descriptor provides an improvement over the state-of-the-art linear classifiers for pedestrian detection, as well for many other datasets (Section 8).
- These techniques can be applied generally to settings where evaluation of weighted additive kernels is required, including kernel PCA, kernel LDA, and kernelized regression, kernelized k-means, as well as efficient training (Section 9).

2 Support Vector Machines

We begin with a review of support vector machines for classification. Given labeled training data of the form $\{(y_i, \mathbf{x}_i)\}_{i=1}^N$, with $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \mathbb{R}^n$, we use a C-SVM formulation [9]. For the linear case, the algorithm finds a hyperplane which best separates the data by minimizing:

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=i}^{N} \xi_i, \tag{1}$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, where $C > 0$, is the tradeoff between regularization and constraint violation. For a kernel on data points, $K(\mathbf{x}, \mathbf{z}) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, that is the inner product, $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$, in an unrealized, possibly high-dimensional, feature space, one can obtain the same by maximizing the dual formulation:

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

$$\textit{subject to:} \quad 0 \leq \alpha_i \leq C \text{ and } \sum \alpha_i y_i = 0. \tag{3}$$

The decision function is $sign(h(\mathbf{x}))$, where:

$$h(\mathbf{x}) = \sum_{l=1}^{m} \alpha_l y_l K(\mathbf{x}, \mathbf{x}_l) + b. \tag{4}$$

Notice that the dual formulation only requires access to the kernel function and not the features $\Phi(.)$, allowing one to solve the formulation in very high-dimensional feature spaces efficiently—also called the *kernel trick*. For clarity, in a slight abuse of notation, the features, $\mathbf{x}_l$: $l \in \{1, 2, \ldots, m\}$, will be referred to as support vectors. Thus, in general, m kernel computations are needed to classify a point with a kernelized SVM and all m support vectors must be stored. Assuming these kernels can be computed in $\mathcal{O}(n)$ time, the overall complexity of the classifier is $\mathcal{O}(mn)$. For linear kernels we can do better because $K(\mathbf{x}, \mathbf{z}) = \mathbf{x} \cdot \mathbf{z}$, so $h(\mathbf{x})$ can be written as $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, where $\mathbf{w} = \sum_{l=1}^{m} \alpha_l y_l \mathbf{x}_l$. As a result, classifying with a linear SVM only requires $\mathcal{O}(n)$ operations and $\mathcal{O}(n)$ memory.

3 Fast Exact IKSVMs

We motivate our discussion using the histogram intersection or the min kernel. Often, similarity between images is obtained by comparing their distribution over low-level features like edge orientations, pixel color values, codebook entries, etc. These distributions could be represented as histograms and a similarity measure like the histogram intersection can be used. The histogram intersection kernel is known to be positive definite [54] for histogram-based features and hence can be used with the standard SVM machinery. This representation is popular for the "bag-of-words" approaches which have led to state-of-the-art results in many object detection and classification tasks.

We first show that it is possible to speed up classification for intersection kernel SVMs. This analysis was first presented in [25] and later independently in our own work [35]. For histogram-based feature vectors $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n_+$, the intersection kernel $K_{\min}(\mathbf{x}, \mathbf{z})$ is defined as:

$$K_{\min}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \min(x_i, z_i) \tag{5}$$

and classification is based on evaluating:

$$h(\mathbf{z}) = \sum_{l=1}^{m} \alpha_l y_l K_{\min}(\mathbf{z}, \mathbf{x}_l) + b \tag{6}$$

$$= \sum_{l=1}^{m} \alpha_l y_l \left(\sum_{i=1}^{n} \min\big(z_i, x_{l,i}\big) \right) + b. \tag{7}$$

The nonlinearity of min prevents us from "collapsing" the weight vector in a similar manner for linear kernels. Thus, the complexity of evaluating $h(\mathbf{x})$ in the standard way is $\mathcal{O}(mn)$. The key property of intersection kernels is that we can exchange the summations in (7) to obtain:

$$h(\mathbf{z}) = \sum_{l=1}^{m} \alpha_l y_l \left(\sum_{i=1}^{n} \min\big(z_i, x_{l,i}\big) \right) + b \tag{8}$$

$$= \sum_{i=1}^{n} \left(\sum_{l=1}^{m} \alpha_l y_l \min\big(z_i, x_{l,i}\big) \right) + b \tag{9}$$

$$= \sum_{i=1}^{n} h_i(z_i) + b. \tag{10}$$

Thus, the overall function $h(\cdot)$ can be rewritten as the sum of 1D functions $h_i(\cdot)$, where:

$$h_i(s) = \sum_{l=1}^{m} \alpha_l y_l \min\big(s, x_{l,i}\big). \tag{11}$$

The complexity of computing each $h_i(s)$ in the naive way is still $\mathcal{O}(m)$ with an overall complexity of computing $h(\mathbf{x})$ still $\mathcal{O}(mn)$. We now show how to compute each h_i in $\mathcal{O}(\log m)$ time.

Consider the functions $h_i(s)$ for a fixed value of i. Let $\bar{x}_{l,i}$ denote the sorted values of $x_{l,i}$ in increasing order with corresponding αs and labels as $\bar{\alpha}_l$ and $\bar{y}_l$. If $s < \bar{x}_{1,i}$ then $h_i(s) = s \sum_l \bar{\alpha}_l = 0$ since $\sum_l \bar{\alpha}_l = 0$. Otherwise, let r be the largest integer such that $\bar{x}_{r,i} \leq s$. Then, we have

$$h_i(s) = \sum_{l=1}^{m} \bar{\alpha}_l \bar{y}_l \min\big(s, \bar{x}_{l,i}\big) \tag{12}$$

$$= \sum_{1 \leq l \leq r} \bar{\alpha}_l \bar{y}_l \bar{x}_{l,i} + s \sum_{r < l \leq m} \bar{\alpha}_l \bar{y}_l \tag{13}$$

$$= A_i(r) + sB_i(r), \tag{14}$$

where we have defined

$$A_i(r) = \sum_{1 \leq l \leq r} \bar{\alpha}_l \bar{y}_l \bar{x}_{l,i}, \tag{15}$$

$$B_i(r) = \sum_{r < l \leq m} \bar{\alpha}_l \bar{y}_l. \tag{16}$$

Equation (14) shows that h_i is piecewise linear. Furthermore, h_i is continuous because

$$\begin{aligned} h_i(\bar{x}_{r+1}) &= A_i(r) + \bar{x}_{r+1} B_i(r) \\ &= A_i(r+1) + \bar{x}_{r+1} B_i(r+1). \end{aligned}$$

Notice that the functions A_i and B_i are independent of the input data and depend only on the support vectors and α. Thus, if we precompute them, then $h_i(s)$ can be computed by first finding r, the position of s in the sorted list $\bar{x}_{l,i}$ using binary search and linearly interpolating between $h_i(\bar{x}_r)$ and $h_i(\bar{x}_{r+1})$. This requires storing the $\bar{x}_l$ as well as the $h_i(\bar{x}_l)$ or twice the storage of the standard implementation. **Thus, the runtime complexity of computing $h(\mathbf{x})$ is $\mathcal{O}(n \log m)$ as opposed to $\mathcal{O}(nm)$, a speedup of $\mathcal{O}(m/\log m)$.** This can be significant if the number of support vectors is large.

4 Approximate Additive Kernel SVMs

It is possible to compute approximate versions of the classifier even faster. Traditional function approximation quickly breaks down as the number of dimension increase. However, for the intersection kernel SVMs we have shown that the final classifier can be represented as a sum of 1D functions. As long as the kernel is "additive," i.e., the overall kernel $K(\mathbf{x}, \mathbf{y})$ can be written as

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} K_i(x_i, y_i), \tag{17}$$

the resulting kernel SVM classifier is also additive, i.e., $h(\mathbf{s})$ can be written as

$$h(\mathbf{s}) = \sum_{i=1}^{n} h_i(s_i) + b, \tag{18}$$

where

$$h_i(s_i) = \sum_{l=1}^{m} \alpha_l y_l K_i(s_i, x_{l,i}) \tag{19}$$

and $x_{l,i}$ denotes the ith dimension of the lth support vector.

This decomposition allows us to approximate the final classifier by approximating each dimension independently. The simplest of these is a piecewise polynomial approximation in which we represent the function in each dimension

as a piecewise polynomial function using b sections, each of degree k. This requires $b \times (k+1)$ floating points per dimension. Classification requires table lookup followed by the evaluation of a k degree polynomial, which requires $2(k+1)$ floating-point operations using Euler's method. Two special cases are the piecewise constant and piecewise linear approximations corresponding to degree $k=0$ and $k=1$, respectively. In our experiments we restrict ourselves to these cases as one can approximate any function arbitrary well. The final classifier corresponds to a lookup table of size $m \times (b+1)$. **The overall complexity of the classifier then is $\mathcal{O}(2(k+1)n)$—essentially the same as that of a linear SVM classifier**.

In our MATLAB/C++ implementation the speed of the piecewise linear approximations is about $5.5\times$ slower than the linear classifier. The more expensive table lookups can be avoided by rewriting the piecewise linear interpolation as a dot product of a dense vector of function values and a sparse vector indicating the bin indices, for e.g., see [34] or [45]. These implementations are essentially as fast as the linear classification method, especially when there are a large number of classes and the encoding time can be amortized over the number of classes.

Although these 1D functions can be precomputed once for each classifier—this could become a bottleneck if the number of classes are large or if the classifier needs to be updated often, for example, during training. To approximate these 1D functions using a piecewise linear approximation, one has to sample these functions at a fixed set of points. The complexity of evaluating these 1D functions $h_i(s_i) = \sum_{l=1}^{m} \alpha_l y_l K_i(s_i, x_{l,i})$ at b locations is $\mathcal{O}(bm)$. When b is large, i.e., $b \gg \log m$, for the intersection kernel one can sample these functions faster using the exact IKSVM evaluation presented in Section 3 in $\mathcal{O}((m+b)\log m)$ time, making it the approach of choice for certain applications.

5 Additive Kernels in Computer Vision

We identify several naturally arising additive kernels in computer vision applications, though we note that variants of these kernels also arise in natural language processing, such as text classification, etc. There are two important classes of additive kernels used in the computer vision, which we describe next.

5.1 Comparing Histograms

Often similarity between images is obtained by comparing their distribution over low-level features like edge orientations, pixel color values, codebook entries, textures, etc. These distributions are typically represented as histograms and a similarity measure like the histogram intersection or the negative χ^2 or l_2 distance is used. Both the histogram intersection kernel [54, and the χ^2 kernels are known to be positive definite for histogram-based features and hence can be used with the standard SVM machinery. See [3], [41] for a proof that the histogram intersection kernel and its variants are positive definite and [2] for a proof for the χ^2 kernel.

The histogram intersection kernel, $K_{\min}$, and the χ^2 kernel, K_{χ^2}, for normalized histograms are defined as follows:

$$K_{\min}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} \min(x_i, z_i), \quad K_{\chi^2}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} \frac{2x_i z_i}{x_i + z_i}. \quad (20)$$

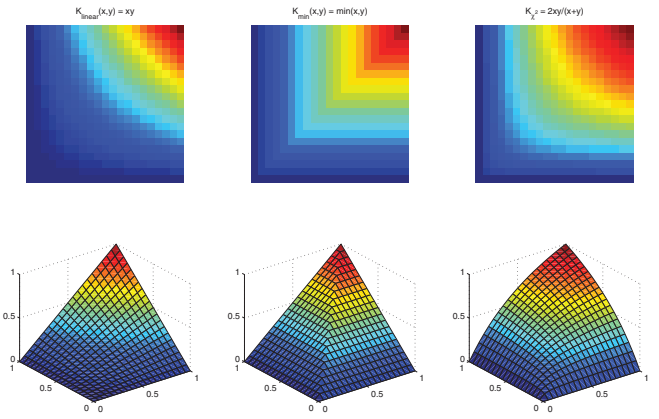

Fig. 2. Visualization of linear, intersection, and χ^2 kernels for 1D features. One can see that the χ^2 kernel is a smoother version of the intersection kernel and is twice differentiable on the interior.

Fig. 2 visualizes these additive kernels. We also note that the intersection kernel is conditionally positive definite for all features and hence can be used with SVMs even when the features are not histograms, i.e., they need not be positive or normalized. For proof, see paper [34]. A special case worth mentioning is the generalized histogram intersection kernel [3] defined by:

$$K(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} \min\left(|x_i|^{\beta}, |z_i|^{\beta}\right). \quad (21)$$

This is known to be positive definite for all $\beta > 0$. Chappelle et al. [7] observe that this remapping of the histogram bin values by $x \rightarrow x^{\beta}$ improves the performance of linear kernel SVMs to become comparable to RBF kernels on an image classification task over the Corel Stock Photo Collection. Simply square-rooting the features with linear kernel, which is also called the Bhattacharyya kernel, has also been shown to provide significant improvements when used with "bag-of-words" style features for various image classification and detection tasks [59], [45]. This representation also arises in a text classification setting where the histograms represent counts of words in a document.

5.2 Approximate Correspondences

Another class of additive kernels is based on the matching sets of features between images. Two popular variants are the *pyramid match* and the *spatial pyramid match* kernels. We describe each of them briefly.

Pyramid match kernel. Introduced by Grauman and Darell [20], [22], who proposed a way to measure similarity between sets of features using partial correspondences between the elements in the sets. The similarity measure reduces to a weighted histogram intersection of features computed in a multiresolution histogram pyramid; hence the name. This approach builds on Indyk and Thaper's [27] approximation to matching costs using l_1 embeddings. An attractive feature of this method is that the matching has linear time complexity in the feature dimension and naturally forms a Mercer kernel, which enables it to be used with discriminative learning frameworks like kernel SVMs. This kernel has been used in various vision tasks like content-based image retrieval, pose estimation, unsupervised category discovery [21], and image classification. This

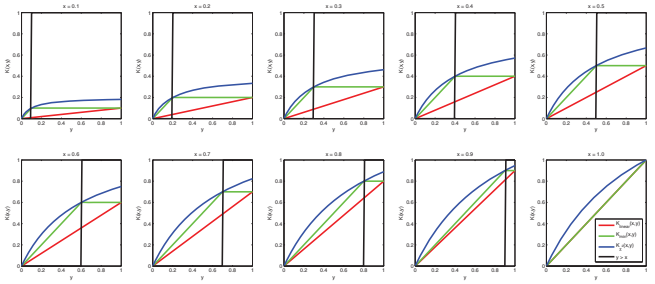

Fig. 3. Visualization of the basis functions of linear, intersection, χ^2 kernels, and decision stumps for 1D features.

kernel is additive because the overall kernel is simply a weighted histogram intersection.

Spatial pyramid match kernel. Lazebnik et al. [32] introduced a similarity based on approximate global geometric correspondence of local features of images. Instead of a global histogram of features one creates histograms of features over increasingly fine subregions of the image in a "spatial pyramid" representation. Like the pyramid match kernel, the spatial matching is now approximated by the weighted histogram intersection of the multiresolution spatial pyramid. This remarkably simple and computationally efficient extension of an orderless bag-of-features has proven to be extremely useful, and has become a standard baseline for various tasks which require image to image similarity like object detection, image classification, pose estimation, action recognition, etc. Many state-of-the-art object detection and image classification results on the PASCAL Visual Object Challenge [12], ImageNet [28], and TRECVID [53] challenge are based on variants of the kernel where the underlying features change. Nevertheless, this kernel is also additive as the overall kernel is once again a weighted histogram intersection of the underlying features.

6 Learning Additive Classifiers

Additive classifiers are based on functions of the form:

$$f(\mathbf{x}) = \sum_{i=1} f_i(x_i), \tag{22}$$

i.e., the overall function f is a sum of 1D functions. Additive functions were popularized by Hastie and Tibshirani [23] for fitting statistics of data. Linear classifiers are the simplest additive classifiers, where each $f_i(x_i) = w_i x_i$. By allowing arbitrary f_i, additive models can provide better fits to the training data than linear models. Our key insight in Section 4, was to observe that if the kernel K is additive, then the learned SVM classifier is also additive. Thus, the standard SVM training machinery provides an efficient way to train additive classifiers compared to the traditional *backfitting* algorithm [18]. Additive classifiers also arise in boosting when the weak-learners are functions of one dimension, for example, decision stumps, $(x_i > c)$. Hence, the standard AdaBoost algorithm [48] is yet another way of training additive classifiers.

We now show that the additive classifiers based on histogram intersection kernel are general, i.e., can represent any additive function on the input features as a linear combination of intersection kernel of the features, as shown by the next theorem.

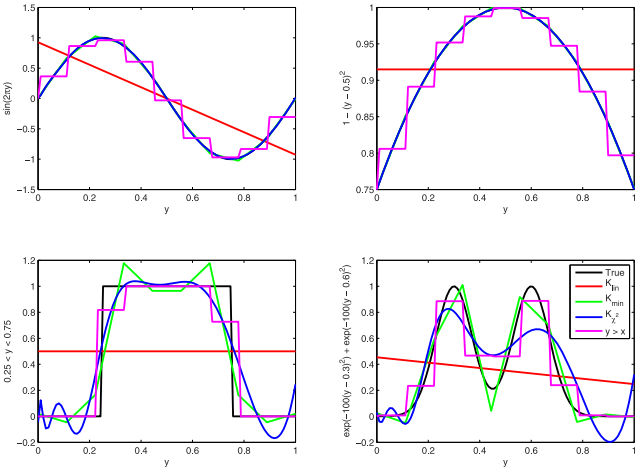

Fig. 4. Approximations of various 1D functions by linear, intersection, χ^2 kernels, and decision stumps as the basis functions.

Theorem 6.1. *Let* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ *be points in* $\mathbb{R}^d \geq 0$ *and* $f(\mathbf{x}_i) = f_1(x_{i,1}) + f_2(x_{i,2}) + \cdots + f_d(x_{i,d})$ *be an additive function, where* $x_{i,j}$ *denotes the value of the* j*th dimension of the* i*th point. Then, there exists* $\alpha_1, \alpha_2, \ldots, \alpha_n$ *such that* $f(\mathbf{x}_i) = \sum_j \alpha_j K_{\min}(\mathbf{x}_i, \mathbf{x}_j)$, $\forall i = 1, 2 \ldots, n$.

Proof. We prove this by showing that there exists a weight vector $\mathbf{w}$, in the Reproducing Kernel Hilbert Space (RKHS) of the intersection kernel, $K_{\min}$, such that $\mathbf{w} \cdot \phi(\mathbf{x}_i) = f(\mathbf{x}_i)$. First we show that there is a weight vector w_k for each f_k such that $w_k \cdot \phi(x_{j,k}) = f_k(x_{j,k})$. This follows immediately from the fact that the gram matrix G^k consisting of entries $G^k_{ij} = \min(x_{i,k}, x_{j,k})$ is full rank for unique $x_{i,k}$, and the system of equations $\alpha G^k = f^k$ has a solution (if the values are not unique, one can remove the repeated entries). Since the overall function is additive, we can obtain the weight vector $\mathbf{w}$ with the required property by stacking the weight vectors, w_k, from each dimension. Thus, by representer theorem, there exists α such that

$$\mathbf{w} \cdot \phi(\mathbf{x}_i) = \sum_j \alpha_j K_{\min}(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i).$$

□

Note that the α is shared across dimensions and this proof may be applied to any additive kernel which satisfies the property that the kernel in each dimension is full rank, for example, the χ^2 kernel.

Thus, the SVM classifier represents the overall function as a linear combination of kernel functions in each dimension. The 1D functions $K_i(s_i, x_{l,i})$ for a fixed value of $x_{l,i}$ can be thought of as a basis function for each dimension of the classifier. Fig. 3 shows these basis functions for the intersection and χ^2 kernels. Fig. 4 shows several 1D functions approximated by a linear combination of 10 basis functions centered at $0.1, 0.2, \ldots, 1.0$ and a constant. The linear combination coefficients were found using linear least-squares regression. The decision stumps $(x_i > c)$ give us a piecewise constant approximation, while the histogram intersection gives a piecewise linear approximation and χ^2 kernel gives smoother polynomial-like approximation.

Compared to the linear case, kernels like the intersection, χ^2 kernel, and decision stumps are able to approximate these functions much better.

7 Previous Work

There are several approaches for speeding up classification using kernel SVM classifiers, which we briefly discuss next.

7.1 Approximate Kernel SVMs

For the histogram intersection kernel, Herbster [25] first proposed the fast evaluation algorithm we presented in Section 3. In our earlier work [35] we independently proposed the same method for exact classification along with the approximate method described in Section 4, which is more general and applies to arbitrary additive kernels. Recently, Rahimi and Recht [46] proposed embeddings that approximate shift-invariant kernels, i.e., $K(\mathbf{x}, \mathbf{y}) = f(|\mathbf{x} - \mathbf{y}|)$, using a feature map Φ such that $K(\mathbf{x}, \mathbf{y}) \sim \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$. Based on this analysis and our own work [34], [35], Vedaldi and Zisserman [59] proposed embeddings which approximate a class of additive kernels that are "homogeneous." This allows one to use the explicit form of the classifier $f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x})$ instead of the kernelized version, which can be more efficient in some settings. We discuss some of these methods in Section 9.

However, during classification for additive kernels, the piecewise linear approximation we proposed can be much faster. To see this, observe that the piecewise linear approximation can be written as a dot product of a weight vector corresponding to the values of the function sampled at uniformly spaced points, with a sparse vector corresponding to the projection of the data onto a uniformly spaced linear B-Spline basis centered at these points (also see [34]). In this representation, evaluating the classifier requires only two multiplications and one addition per dimension, which can be much smaller compared to the approximate embeddings of [59].

Another line of approach applicable to Gaussian kernels is the work of Yang et al. [66] who use the fast Gauss transform to build efficient classifiers—however, this is applicable when the feature dimension is very small, typically less than 10.

7.2 Reduced Set Methods

For general kernels, a class of methods known as "reduced set methods" approximates the classifier by constructing representations using a small subset of data points, typically much smaller than the number of support vectors. These sets of points can be the set of input points themselves, as in the work of Burges [6], Osuna and Girosi [43], where the most representative support vectors are kept as a postprocessing step. Instead of having a single approximation, one can have a series of approximations with more and more points to obtain a cascade of classifiers, an idea which has been used in [47] to build fast face detectors. Another class of methods builds classifiers by having a regularizer in the optimization function which encourages sparseness, (e.g., l_1-*norm* on the alphas) or picks support vectors in a greedy manner till a stopping criterion is met [30]. These methods may be able to reduce the number of support vectors by an order of magnitude, but are still significantly slower than a linear SVM. Often this come at the expense of classification accuracy. Thus, these approaches are not competitive when the kernel is additive compared to our approach.

7.3 Coarse to Fine Methods

The coarse to fine approach for speeding up the classification is popular in many real-time vision applications. Simpler features and classifiers are used to reject easy examples quickly in a cascade. This idea has been applied to face [24], [4] and pedestrian detection [64] to achieve an order of magnitude speedup in the overall detection time. Methods like branch and bound [31], context [26], bottom-up regions [56], Hough transformation [37], [33], etc., improve efficiency by reducing the number of classifier evaluations. This paper improves the efficiency of the underlying discriminative classifier, allowing more powerful classifiers to be evaluated exponentially faster—in practice up to several thousand times faster than naive implementations and entirely complementary to the techniques mentioned for reducing the number of classifier evaluations.

8 Experimental Results

Since its introduction, our ideas for efficiently computing weighted combination for additive kernels has been applied to many applications like image-classification on the Caltech-101 [15], PASCAL Visual Object Challenge [12], handwritten digits [36], video retrieval (TRECVID [53]), near-duplicate image detection [52], pedestrian detection frameworks combining static image features and optical flow [62], efficient classifiers for training large scale data [34], [59], [63], etc. We summarize some of these applications in Section 10.

We present experiments on several image classification and detection datasets and compare the performance of linear intersection as well as a nonlinear kernel, such as radial basis or polynomial kernel. We also report the speedup obtained by the piecewise linear approximation compared to the naive method of evaluating the classifier. Table 1 contains a summary of our results. The piecewise linear approximations are as accurate as the exact additive classifier, with about 100 pieces on all datasets. **On various datasets, the intersection kernel SVM is significantly better than the linear SVM and often comparable to rbf-kernel SVM while offering up to three orders of magnitude speedup**. The details of each dataset and the features are presented below.

8.1 Toy Example: Learning a Circle

We illustrate the additive kernel approximation using a toy example. The data are generated by sampling points from a 2D Gaussian and all points within a certain radius of the center belong to one class and the points outside belong to the other class, as seen in Fig. 5 (top-left).

A linear classifier works poorly in this case as no 2D line can separate the points well. However, the intersection kernel SVM is able to achieve an accuracy of 99.10 percent on this data. This is because it is able to approximate the circle, which is an additive function ($x^2 + y^2 \leq r$), using two 1D curves, x^2 and y^2. Fig. 5 shows the learned classifier represented with varying number of bins using a piecewise linear approximation, as well as the classification accuracy as a function of the number of approximation bins. The accuracy saturates with 10 bins. On more realistic datasets,

TABLE 1
Summary of Our Results

Dataset	Features	Measure	Linear SVM	IKSVM	Kernel SVM	Kernel Type
Toy Dataset	Raw	Accuracy	51.9%	99.10% (22×)	99.50%	$rbf, \gamma = 0.5$
MNIST Digits	OV-SPHOG	Error Rate	1.44%	0.77% (1200×)	0.56%	$poly, d = 5$
USPS Digits	Raw Pixels	Error Rate	11.3%	8.7% (24×)	4.0%	$poly, d = 3$
	OV-SPHOG	Error Rate	3.4%	3.4% (26×)	3.2%	$poly, d = 5$
INRIA Pedestrian	SPHOG	Recall at 2 FPPI	43.12%	86.59% (2594×)	-	
DC Pedestrians	SPHOG	Accuracy	$72.19 \pm 4.40\%$	$89.03 \pm 1.39\%$ (2253×)	$88.13 \pm 1.43\%$	$rbf, \gamma = 175$
Caltech 101, 15 examples	SPHOG	Accuracy	$38.79 \pm 0.94\%$	$50.10 \pm 0.65\%$ (37×)	$44.27 \pm 1.45\%$	$rbf, \gamma = 250$
30 examples	SPHOG	Accuracy	$44.33 \pm 1.33\%$	$56.59 \pm 0.77\%$ (62×)	$50.13 \pm 1.19\%$	$rbf, \gamma = 250$
UIUC Cars (Single Scale)	SPHOG	Precision at EER	89.8%	98.5% (65×)	93.0%	$rbf, \gamma = 2.0$

We show the performance using a linear, intersection, and nonlinear kernel as well as the speedup obtained by a piecewise linear approximation of the intersection kernel classifier on each dataset. The rbf kernel is defined as $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma(\mathbf{x} - \mathbf{y})^2)$ and the $poly$ kernel of degree d, is defined as $K(\mathbf{x}, \mathbf{y}) = (1 + \gamma(\mathbf{x} \cdot \mathbf{y}))^d$. All the kernel hyperparameters were set using cross validation.

the number of bins required for a good approximation depends on the smoothness of the underlying function, but empirically, 100 bins were sufficient in all our experiments.

8.2 MNIST and USPS Digits

The MNIST dataset[1] was introduced by LeCun and Cortes and contains 60,000 examples of digits 0-9 for training and 10,000 examples for testing. As before, we construct features based on histograms over oriented responses computed by convolving the image with a Gaussian derivative filter with $\sigma = 2$ and bin the response in 12 orientations. The images in this dataset are 28×28 pixels and we collect histograms over blocks of sizes 28×28, 14×14, 7×7, and 4×4 pixels. We also found that adding overlapping blocks which overlap the block size by half improves performance at the expense of increasing the feature vector dimension by a factor of about four. This is similar in spirit to the overlapping blocks in the HOG descriptor in the pedestrian detector of [10]. These features with an IKSVM classifier achieve an error rate of 0.79 percent, compared to an error rate of 1.44 percent using linear and 0.56 percent using polynomial kernel. Similar features and IKSVM achieves an error rate of 3.4 percent on the much harder USPS dataset. We refer the readers to [36] for a complete set of experiments for the task of handwritten digit classification. Fig. 6 shows the errors made by our digit recognition system on the MNIST dataset.

A key advantage is that the resulting IKSVM classifier is very fast. The estimated number of multiply-add operations required by the linear SVM is about 40K, while the intersection kernel requires about 125K operations, including the time to compute the features. This is significantly less than the about 14 million operations required by a polynomial kernel SVM reported in the work of Decoste and Schölkopf [11]. The reduced set methods [5] (1.0 percent error) require approximately 650K operations, while the neural network methods like LeNet5 (0.9 percent error) requires 350K and the boosted LeNet4 (0.7 percent error) requires 450K operations. For a small cost for computing features we are able to achieve competitive performance while at the same time being faster at both training and test time.

1. http://yann.lecun.com/exdb/mnist/.

8.3 INRIA Pedestrians

The INRIA pedestrian dataset [10] was introduced as an alternate to the existing pedestrian datasets (e.g., MIT Pedestrian Data set) and is significantly harder because of the wide variety of articulated poses, variable appearance/clothing, illumination changes, and complex backgrounds. Linear kernel SVMs with Histograms of Oriented Gradients (HOG) features achieve high accuracy and speed on this dataset [10]. We use the multiscale HOG features introduced in [35] and train a intersection kernel SVM on these features. The single scale HOG used in the original paper [10] when used with IKSVM provides small improvements over the linear kernel, similar to those observed by using the rbf-kernel. We also found that the HOG with l_1-normalization of the gradient-based features works better with the intersection kernel. The multiscale HOG, however, outperforms l_1-normalized HOG. Results are shown in

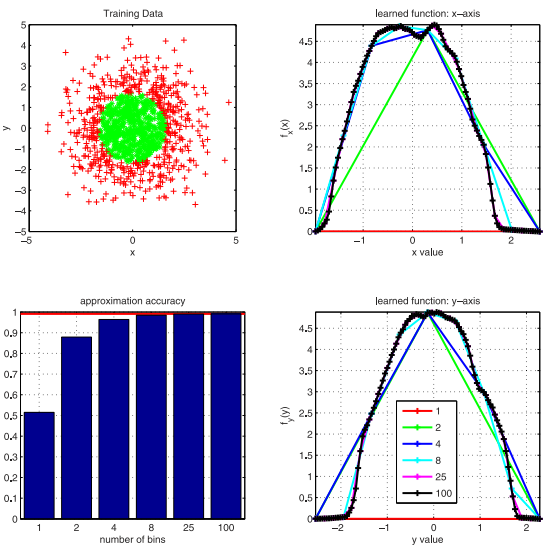

Fig. 5. Toy example. Top left: The training data. Bottom left: Accuracy of the learned classifier approximated by a piecewise linear function of varying number of bins. Top right and top left: Learned functions on the x and y dimensions, respectively, as well as the piecewise linear approximations using a varying number of bins.

Fig. 6. All the errors made by the classifier on the MNIST dataset. On each image, $a \rightarrow b$ means that the digit a was misclassified as b.

Table 2 using 100 bin approximation. Fig. 7 shows sample detections on this dataset.

8.4 Daimler Chrysler Pedestrians

We use the Daimler Chrysler pedestrian benchmark dataset, created by Munder and Gavrila [39]. The dataset is split into five disjoint sets, three for training and two for testing. Each training set has 5,000 positive and negative examples each, while each test set has 4,900 positive and negative examples each. We report results by training on two out of three training sets at a time and testing on each of the test sets to obtain six train-test splits. Due to the small size of the images (18×36), we compute the multilevel features with only three levels ($L = 3$) of pyramid with

TABLE 2
Detection Rate at 2 FPPI on the INRIA Person Dataset

Classification Method	Detection Rate (2 FPPI)	Speedup
Linear SVM	43.12 %	-
IKSVM (binary search)	86.59 %	**473**×
IKSVM (piecewise linear)	86.59 %	**2594**×
IKSVM (piecewise constant)	86.59 %	**3098**×
Dalal & Triggs [10]	79.63 %	-
Dalal & Triggs [10]*	82.51 %	-

The last run of [10] is obtained by running the detector using a finer "scaleratio" of 1.05 between successive layers of the image pyramid instead of the default 1.1.

TABLE 3
Accuracy on the Daimler-Crysler Pedestrian Dataset

Classification Method	Accuracy(%)	Speedup
Linear SVM	72.19 ± 4.40	-
IKSVM (binary search)	89.06 ± 1.42	**485**×
IKSVM (piecewise linear)	89.03 ± 1.39	**2253**×
IKSVM (piecewise constant)	88.83 ± 1.39	**3100**×
RBF-SVM	88.85 ± 1.13	–

cell sizes 18×18, 6×6, and 3×3 at levels 1, 2, and 3, respectively. The block normalization is done with a cell size of $w_n \times h_n = 18 \times 18$. The features at level l are weighted by a factor $c_l = 1/4^{(L-l)}$ to obtain a 656-dimensional vector which is used to train an IKSVM classifier.

The classification results using the exact methods and approximations are shown in Table 3. Our results are comparable to the best results for this task [39]. The IKSVM classifier is comparable in accuracy to the rbf-kernel SVM, and significantly better than the linear SVM. The speedups obtained for this task are significant due to large number of support vectors in each classifier. The piecewise linear with 30 bins is about **2,000**× faster and requires **200**× less memory, with no loss in classification accuracy. The piecewise constant approximation, on the other hand, requires about 100 bins for similar accuracies and is even faster.

Our unoptimized MATLAB implementation for computing the features takes about 17 ms per image and the time for classification (0.02 ms) is negligible compared to

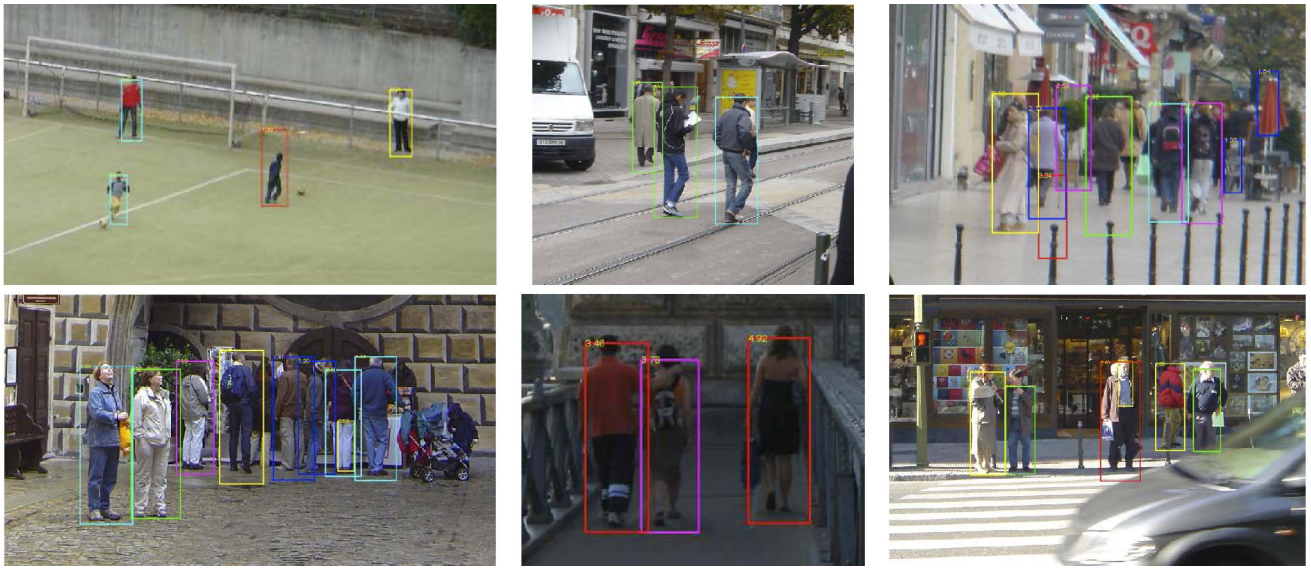

Fig. 7. Sample pedestrian detections on the INRIA person dataset using the spHOG + IKSVM classifier.

Fig. 8. Top row, False negatives and, bottom row, false positives on the Daimler-Chrysler pedestrian dataset.

TABLE 4
Classification Accuracy of Various Methods on the Caltech-101 Dataset Using 15 and 30 Training Examples per Category

	15 examples, 115.2 ± 15 SVs		30 examples, 185.0 ± 26 SVs	
Classification Method	**Accuracy(%)**	**Speedup**	**Accuracy(%)**	**Speedup**
Linear SVM	38.79 ± 0.94	-	44.33 ± 1.33	-
IKSVM (binary search)	50.15 ± 0.61	$\mathbf{11}\times$	56.49 ± 0.78	$\mathbf{17}\times$
IKSVM (piecewise linear)	50.10 ± 0.65	$\mathbf{37}\times$	56.59 ± 0.77	$\mathbf{62}\times$
IKSVM (piecewise constant)	49.83 ± 0.62	$\mathbf{45}\times$	56.11 ± 0.94	$\mathbf{76}\times$

The piecewise linear classifiers are up to $60\times$ faster without loss in accuracy over the exact method.

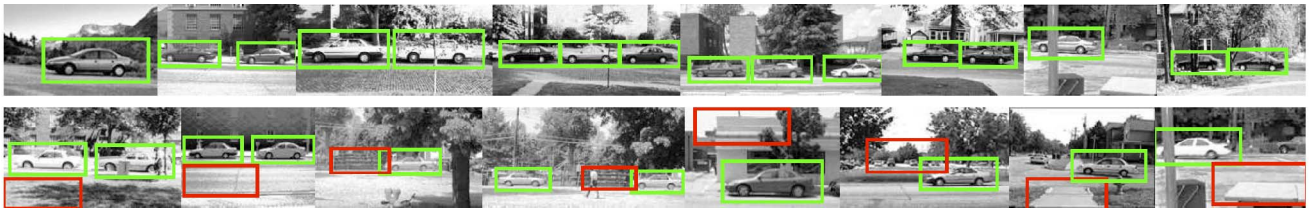

Fig. 9. Example detections (green) and misdetections (red) of the detector on the UIUC cars dataset.

this. Compared to the 250 ms required by the cascaded SVM-based classifiers of Munder and Gavrila [39], our pipeline is $15\times$ faster. Fig. 8 shows some of the errors made by our classifier.

8.5 Caltech 101

Our next set of experiments are on Caltech-101 [15]. The aim here is to show that existing methods can be made significantly faster, even when the number of support vectors in each classifier is small. We use the framework of Lazebnik et al.[32] and use our own implementation of their "weak features" and achieve an accuracy of 56.49 percent (compared to their 54 percent), with 30 training and test examples per class and one-versus-all classifiers based on IKSVM. The performance of a linear SVM using the same features is about 44.33 percent, while that of an rbf kernel is 50.13 percent. The IKSVM classifiers on average have 185 support vectors and a piecewise linear approximation with 60 bins is $62\times$ faster and the piecewise constant approximation is $76\times$ faster than a standard implementation, with no loss in accuracy (see Table 4).

It is interesting to note the performance of one-versus-one classifiers as they are faster to train. With 15 training and 50 test examples per category, one-versus-one classifiers give an accuracy of 47.43 ± 0.37 for intersection, compared to 39.58 ± 0.78 for linear kernel, with five-fold cross validation. Increasing with number of training examples to 30 improves the performance to 53.80 ± 2.43 for intersection kernel compared to 45.66 ± 2.63 for linear kernel.

8.6 UIUC Cars

This dataset was collected at UIUC [1] and contains images of side views of cars. The training set consists of 550 car and 500 noncar images. We test our methods on the single scale image test set, which contains 170 images with 200 cars. The images are of different sizes themselves but contain cars of approximately the same scale as in the training images. Results are shown in Table 5. Once again the IKSVM classifier outperforms both the linear and the rbf kernel SVM and is comparable to the state of the art. Fig. 9 shows some of the detections and misdetections on this dataset.

8.7 Comparison to the "Square-Root" Kernel

As has been noted earlier [7] and recently by several others [45], [60], simply square-rooting the features, also known as the "Bhattacharyya" kernel, can improve the performance of linear classifiers significantly. This method has the advantage that it requires no special machinery on top of linear training and testing algorithms. For example, on the Caltech-101 dataset, Vedaldi and Zisserman [60] observe a

TABLE 5
Performance at Equal Error Rate on the UIUC Cars Dataset

Classification Method	**Performance(%)**	**Speedup**
Linear SVM	89.8	-
IKSVM (binary search)	98.5	$\mathbf{23}\times$
IKSVM (piecewise linear)	98.5	$\mathbf{65}\times$
IKSVM (piecewise constant)	98.5	$\mathbf{83}\times$
RBF-SVM	93.0	-
Agarwal & Roth [1]	79.0	-
Garg *et al.* [19]	88.0	-
Fergus *et al.* [17]	88.5	-
ISM [33]	97.5	-
Mutch & Lowe [40]	99.6	-
Lampert *et al.* [31]	98.5	-

12 percent improvement over linear SVM by square-rooting the features and an additional 3 percent improvement using an additive kernel. On the PASCAL VOC 2007 dataset, [45] observe a 4.8 percent improvement using "square-root" kernel compared to linear, and an additional 1.6 percent improvement using the intersection kernel and a similar trend on the ImageNet dataset.

9 Extensions and Applications

The techniques and analysis we propose can be applied generally to settings where evaluation of weighted additive kernels is required. This includes kernelized versions of PCA, LDA, regression, and k-means. In addition one can use our method to speed up the inner loop of training SVM classifiers. We describe some of these extensions next.

9.1 Efficient Training Algorithms

Though the focus of this paper is on efficient evaluation of the classifier, our method can be used for training the classifier itself. Although, for an application classification, speed is the most important factor, training these classifiers can become a bottleneck. Several training algorithms use classification as an inner loop to identify misclassified examples to update the classifier. When the number of training examples is very large, for example while training object detectors, one often employs bootstrapping to train successive classifiers and collect "hard" examples by running the classifiers on large amounts of data. In such scenarios, the overall pipeline can be sped up using the techniques proposed in this paper.

Recent advancements in the learning community has lead to training algorithms for linear SVMs which scale linearly with the training data (LIBLINEAR [13], PEGASOS [51], SVM^{perf} [29]). However, for general kernel SVMs the training times can still be very high. Often the implicit RKHS, denoted by $\Phi(x)$, such that $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, of the kernel is very high dimensional (possibly infinite), which makes it impractical to use the linear SVM techniques directly in the RKHS.

It is possible to construct low-dimensional or sparse embeddings which preserve the dot products in kernel space *approximately* ($K(\mathbf{x}, \mathbf{y}) \sim \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$) to leverage fast linear SVM training algorithms. For shift invariant kernels this has been addressed in [46]. These kernels have a special form $K(\mathbf{x}, \mathbf{y}) = f(|\mathbf{x} - \mathbf{y}|)$. The histogram intersection and χ^2 kernels are not shift invariant, but the approximate versions of the classifier, namely, the piecewise constant/linear classifiers, can be thought of as embeddings that preserve the kernel dot product approximately. In particular, [34] describes a sparse embedding which has at most twice the number of nonzero entires as the original feature space, which allows efficient linear training techniques to be used. More recently, Vedaldi and Zisserman [59] proposed alternate embeddings which are low dimensional for additive kernels.

Another approach is to use the efficient classification techniques internally to speed up the training process [63]. The performance of these approximate classifiers are often identical to that of the trained kernel SVMs while taking a fraction of the training time, as shown by the experiments in [34], [59]. On several benchmark datasets related to image classification and object detection, these techniques provide a significant improvement in accuracy over linear SVMs while paying only a small cost in training and test time, which renders this paradigm of practical importance.

9.2 Classifier Cascades

One can order the class of kernels based on the representation power, $Linear \subset Additive \subset General$, to create a cascade of detectors where a linear classifier is used to first discard the easy examples and the rest are then passed on to an additive kernel classifier and so on. This idea has been used for object detection by Vedaldi et al. [58] to build cascades of object detectors based on multiple kernels derived from color, texture, and shape features and was the top performing method on the PASCAL 2008 VOC object detection challenge.

9.3 Additive Kernel Methods

Finally, the additive kernel approximation ideas apply generally to various other tasks like regression, clustering, ranking, principal components, etc. Methods like kernel PCA, LDA, Gaussian processes, etc., can benefit from our analysis to vastly speed up the computation times as well as reduce the runtime memory and storage requirements. We describe some of these in this section.

9.3.1 Kernel Regression/Gaussian Processes

This is analogous to the classification case, except the outputs y_i are real valued instead of $\{+1, -1\}$ for the classification case. The optimization problem often involves minimizing something like the squared-error between the prediction and the true value [50]. Similarly to the classification case, the final regression function is of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b. \tag{23}$$

9.3.2 Kernel PCA/LDA

Dimensionality reduction techniques like Principal Component Analysis (PCA) have been extended to arbitrary Hilbert spaces using the kernel trick [49]. These, however, come at a great increase in computational and memory cost. For an arbitrary kernel K, the projection of a new data point $\mathbf{x}$ to the nth principal component takes the form [50]:

$$c_n(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i^n K(\mathbf{x}_i, \mathbf{x}), \tag{24}$$

where α^n is the nth eigenvector of the kernel matrix $K_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$. For additive kernels one can use our proposed techniques to both represent these principal components compactly as well as compute the projections of a new data point onto the principal components efficiently. This idea has been used in [45] to compute low-dimensional representations of features for image classification.

Fisher Discriminant Analysis or Linear Discriminant Analysis (LDA) in the kernel space has a similar form as one of the projection vectors in PCA, i.e., $\sum_{i=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{x})$. The α vector is the solution to the kernelized version of the optimization problem in LDA. We refer the reader to [38] for details.

9.3.3 Kernel Codebooks

Kernelized similarity can be used for unsupervised clustering to construct codebooks used bag-of-words models for visual recognition. Methods like k-means or Gaussian mixture models represent cluster centers as a weighted

combination of the points within the cluster and evaluating the similarity to a point involves the weighted combination of kernel similarity to points within the cluster, a step which can benefit from our analysis. In [65], the authors demonstrate that histogram intersection kernel-based codebooks offer better accuracies on several image classification tasks over traditional clustering based on linear kernel.

10 Conclusion

In this paper, we showed that a class of nonlinear kernels called *additive* kernels leads to SVM classifiers which can be approximately evaluated very efficiently. Additive kernels are strictly more general than linear kernels and often lead to significant improvements and our technique brings down the memory and time complexity of classification using additive kernels to only a small constant multiple of that of a linear SVM. Additive kernels are widely used in computer vision and our technique has found widespread applications in many classification/detection tasks.

In addition, our technique has led to efficient training algorithms for additive classifiers and has sped up many applications involving histogram-based comparison, like near duplicate image detection and kernel k-means/PCA/LDA/regression.

Acknowledgments

Subhransu Maji was supported by US Army Research Office MURI W911NF-06-1-0076, US Office of Naval Research MURI N00014-06-1-0734, and a Google fellowship. Alexander C. Berg acknowledges Stony Brook University start-up funding.

References

[1] S. Agarwal and D. Roth, "Learning a Sparse Representation for Object Detection," *Proc. European Conf. Computer Vision,* 2002.

[2] S. Belongie, C. Fowlkes, F. Chung, and J. Malik, "Spectral Partitioning with Indefinite Kernels Using the Nystrom Extension," *Proc. European Conf. Computer Vision,* 2002.

[3] S. Boughorbel, J.-P. Tarel, and N. Boujemaa, "Generalized Histogram Intersection Kernel for Image Recognition," *Proc. IEEE Conf. Image Processing,* 2005.

[4] L. Bourdev and J. Brandt, "Robust Object Detection via Soft Cascade," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[5] C. Burges and B. Schölkopf, "Improving the Accuracy and Speed of Support Vector Machines," *Proc. Neural Information Processing Systems,* 1997.

[6] C.J.C. Burges, "Simplified Support Vector Decision Rules," *Proc. Int'l Conf. Machine Learning,* 1996.

[7] O. Chapelle, P. Haffner, and V. Vapnik, "Support Vector Machines for Histogram-Based Image Classification," *IEEE Trans. Neural Networks,* vol. 10, no. 5, pp. 1055-1064, Sept. 1999.

[8] O. Chum and A. Zisserman, "Presented at Pascal Visual Recognition Challenge Workshop," 2007.

[9] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[10] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[11] D. Decoste and B. Schölkopf, "Training Invariant Support Vector Machines," *Machine Learning,* vol. 46, nos. 1-3, pp. 161-190, 2002.

[12] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int'l J. Computer Vision,* vol. 88, no. 2, pp. 303-338, June 2010.

[13] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A Library for Large Linear Classification," *J. Machine Learning Research,* vol. 9, pp. 1871-1874, 2008.

[14] L. Fei-Fei, R. Fergus, and P. Perona, "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.

[15] L. Fei-Fei, R. Fergus, and P. Perona, "One-Shot Learning of Object Categories," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 4, pp. 594-611, Apr. 2006.

[16] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A Discriminatively Trained Multiscale, Deformable Part Model," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[17] R. Fergus, P. Perona, and A. Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2003.

[18] J. Friedman and W. Stuetzle, "Projection Pursuit Regression," *J. Am. Statistical Assoc.,* vol. 76, pp. 817-823, 1981.

[19] A. Garg, S. Agarwal, and T.S. Huang, "Fusion of Global and Local Information for Object Detection," *Proc. Int'l Conf. Pattern Recognition,* 2002.

[20] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* 2005.

[21] K. Grauman and T. Darrell, "Unsupervised Learning of Categories from Sets of Partially Matching Image Features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[22] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Efficient Learning with Sets of Features," *J. Machine Learning Research,* vol. 8, pp. 725-760, 2007.

[23] T. Hastie and R. Tibshirani, *Generalized Additive Models.* Chapman & Hall/CRC, 1990.

[24] B. Heisele, T. Serre, S. Prentice, and T. Poggio, "Hierarchical Classification and Feature Reduction for Fast Face Detection with Support Vector Machines," *Pattern Recognition,* vol. 36, no. 11, pp. 2007-2017, Sept. 2003.

[25] M. Herbster, "Learning Additive Models Online with Fast Evaluating Kernels," *Proc. 14th Ann. Conf. Computational Learning Theory and Fifth European Conf. Computational Learning,* pp. 444-460, 2001.

[26] D. Hoiem, A. Efros, and M. Hebert, "Putting Objects in Perspective," *Int'l J. Computer Vision,* vol. 80, no. 1, pp. 3-15-15, Oct. 2008.

[27] P. Indyk and N. Thaper, "Fast Image Retrieval via Embeddings," *Proc. Third Int'l Workshop Statistical and Computational Theories of Vision,* 2003.

[28] W.J. Deng, R. Dong, L.-J. Socher, K.L. Li, and L. Fei-Fei, "Imagenet: A Large-Scale Hierarchical Image Database," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2009.

[29] T. Joachims, "Training Linear svms in Linear Time," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2006.

[30] S.S. Keerthi, O. Chapelle, and D. DeCoste, "Building Support Vector Machines with Reduced Classifier Complexity," *J. Machine Learning Research,* vol. 7, pp. 1493-1515, 2006.

[31] C.H. Lampert, M.B. Blaschko, and T. Hofmann, "Beyond Sliding Windows: Object Localization by Efficient Subwindow Search," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[32] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[33] B. Leibe, A. Leonardis, and B. Schiele, "Combined Object Categorization and Segmentation with an Implicit Shape Model," *Proc. European Conf. Computer Vision Workshop Statistical Learning in Computer Vision,* pp. 17-32, 2004.

[34] S. Maji and A.C. Berg, "Max Margin Additive Classifiers for Detection," *Proc. 12th IEEE Int'l Conf. Computer Vision,* 2009.

[35] S. Maji, A.C. Berg, and J. Malik, "Classification Using Intersection Kernel Support Vector Machines is Efficient," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[36] S. Maji and J. Malik, "Fast and Accurate Digit Classification," Technical Report UCB/EECS-2009-159, EECS Department, Univ. of California, Berkeley, Nov. 2009.

[37] S. Maji and J. Malik, "Object Detection Using a Max-Margin Hough Transform," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2009.

[38] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers, "Fisher Discriminant Analysis with Kernels," *Proc. IEEE Signal Processing Soc. Workshop Neural Networks for Signal Processing,* 1999.

[39] S. Munder and D.M. Gavrila, "An Experimental Study on Pedestrian Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 11, pp. 1863-1868, Nov. 2006.

[40] J. Mutch and D.G. Lowe, "Multiclass Object Recognition with Sparse Localized Features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[41] F. Odone, A. Barla, and A. Verri, "Building Kernels from Binary Strings for Image Matching," *IEEE Trans. Image Processing,* vol. 14, no. 2, pp. 169-180, Feb. 2005.

[42] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 1997.

[43] E.E. Osuna and F. Girosi, "Reducing the Run-Time Complexity in Support Vector Machines," *Advances in Kernel Methods: Support Vector Learning,* pp. 271-283, The MIT Press, 1999.

[44] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *Int'l J. Computer Vision,* vol. 38, no. 1, pp. 15-33, 2000.

[45] F. Perronnin, J. Sandnchez, and Y. Liu, "Large-Scale Image Categorization with Explicit Data Embedding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2010.

[46] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," *Proc. Neural Information Processing Systems,* 2007.

[47] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake, "Computationally Efficient Face Detection," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* 2001.

[48] R. Schapire, "A Brief Introduction to Boosting," *Proc. Int'l Joint Conf. Artificial Intelligence,* 1999.

[49] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation,* vol. 10, no. 5, pp. 1299-1319, 1998.

[50] B. Scholkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2001.

[51] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal Estimated Sub-Gradient Solver for SVM," *Proc. Int'l Conf. Machine Learning,* 2007.

[52] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-Time Large Scale Near-Duplicate Web Video Retrieval," *Proc. ACM Int'l Conf. Multimedia,* 2010.

[53] A.F. Smeaton, P. Over, and W. Kraaij, "Evaluation Campaigns and Trecvid," *Proc. Eighth ACM Int'l Workshop Multimedia Information Retrieval,* pp. 321-330, 2006.

[54] M.J. Swain and D.H. Ballard, "Color Indexing," *Int'l J. Computer Vision,* vol. 7, no. 1, pp. 11-32, 1991.

[55] A. Torralba, K. Murphy, and W. Freeman, "Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.

[56] K. Van de Sande, J. Uijlings, T. Gevers, and A. Smeulders, "Segmentation as Selective Search for Object Recognition," *Proc. 13th IEEE Int'l Conf. Computer Vision,* 2011.

[57] M. Varma and D. Ray, "Learning the Discriminative Power-Invariance Trade-Off," *Proc. 11th IEEE Int'l Conf. Computer Vision,* 2007.

[58] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple Kernels for Object Detection," *Proc. 12th IEEE Int'l Conf. Computer Vision,* 2009.

[59] A. Vedaldi and A. Zisserman, "Efficient Additive Kernels via Explicit Feature Maps," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2010.

[60] A. Vedaldi and A. Zisserman, "Efficient Additive Kernels via Explicit Feature Maps," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 34, no. 3, pp. 480-492, Mar. 2012.

[61] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.

[62] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New Features and Insights for Pedestrian Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2010.

[63] G. Wang, D. Hoiem, and D. Forsyth, "Learning Image Similarity from Flickr Groups Using Stochastic Intersection Kernel Machines," *Proc. 12th IEEE Int'l Conf. Computer Vision,* 2009.

[64] W. Zhang, G. Zelinsky, and D. Samaras, "Real-Time Accurate Object Detection Using Multiple Resolutions," *Proc. 11th IEEE Int'l Conf. Computer Vision,* 2007.

[65] J. Wu and J.M. Rehg, "Beyond the Euclidean Distance: Creating Effective Visual Codebooks Using the Histogram Intersection Kernel," *Proc. 12th IEEE Int'l Conf. Computer Vision,* 2009.

[66] C. Yang, R. Duraiswami, N.A. Gumerov, and L. Davis, "Improved Fast Gauss Transform and Efficient Kernel Density Estimation," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* 2003.

[67] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study," *Int'l J. Computer Vision,* vol. 73, no. 2, pp. 213-238, 2007.

Subhransu Maji received the BTech degree in computer science and engineering from the Indian Institute of Technology, Kanpur, in 2006, and the PhD degree in computer science from the University of California, Berkeley, in 2011. He is currently a research assistant professor at TTI Chicago. Earlier he was an intern in Google's Image Search Group and INRIA's LEAR Group, and a visiting researcher at Microsoft Research India and the Johns Hopkins University. His primary interests are in computer vision, with focus on representations and efficient algorithms for visual recognition. He received the medal for the best graduating student in computer science from IIT Kanpur. He was one of the recipients of the Google Graduate Fellowship in 2008. He is a member of the IEEE.

Alexander C. Berg received the BA and MA degrees in mathematics from the Johns Hopkins University, the PhD degree in computer science from the University of California, Berkeley, in 2005. He is currently an assistant professor at Stony Brook University. Prior to that, he was a research scientist at Yahoo! Research and later at Columbia University. His research addresses challenges in visual recognition at all levels, from features for alignment and object recognition, to action recognition in video, to object detection, to high-level semantics of hierarchical classification, attributes, and natural language descriptions of images. Much of this work has a focus on efficient large scale solutions. He is a member of the IEEE.

Jitendra Malik received the BTech degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1980 and the PhD degree in computer science from Stanford University in 1985. In January 1986, he joined the University of California (UC), Berkeley, where he is currently the Arthur J. Chick Professor in the Computer Science Division, Department of Electrical Engineering and Computer Sciences (EECS). He is also on the faculty of the Department of Bioengineering, and is a member of the Cognitive Science and Vision Science groups. During 2002-2004 he served as the chair of the Computer Science Division and during 2004-2006 as the Department Chair of EECS. He serves on the advisory board of Microsoft Research India and on the Governing Body of IIIT Bangalore. His current research interests are in computer vision, computational modeling of human vision, and analysis of biological images. His work has spanned a range of topics in vision, including image segmentation, perceptual grouping, texture, stereopsis, and object recognition with applications to image-based modeling and rendering in computer graphics, intelligent vehicle highway systems, and biological image analysis. He has authored or coauthored more than a 150 research papers on these topics, and graduated 30 PhD students. According to Google Scholar, five of his papers have received more than 1,000 citations each. He is one of ISI's Highly Cited Researchers in Engineering. He received the gold medal for the best graduating student in Electrical Engineering from IIT Kanpur in 1980 and a Presidential Young Investigator Award in 1989. At UC Berkeley, he was selected for the Diane S. McEntyre Award for Excellence in Teaching in 2000, a Miller Research Professorship in 2001, and appointed to be the Arthur J. Chick Professor in 2002. He received the Distinguished Alumnus Award from IIT Kanpur in 2008. He was awarded the Longuet-Higgins Prize for a contribution that has stood the test of time twice, in 2007 and in 2008. He was elected to the National Academy of Engineering in 2011. He is a fellow of the IEEE and the ACM.