Multiple Kernels for Object Detection

Andrea Vedaldi[1] Varun Gulshan[1] Manik Varma[2] Andrew Zisserman[1]

[1]Department of Engineering Science
University of Oxford
{vedaldi,varun,az}@robots.ox.ac.uk

[2]Microsoft Research India
Second Main Road, Sadashiv Nagar, Bangalore 560 080 India
manik@microsoft.com

Abstract

Our objective is to obtain a state-of-the art object category detector by employing a state-of-the-art image classifier to search for the object in all possible image subwindows. We use multiple kernel learning of Varma and Ray (ICCV 2007) to learn an optimal combination of exponential χ^2 kernels, each of which captures a different feature channel. Our features include the distribution of edges, dense and sparse visual words, and feature descriptors at different levels of spatial organization.

Such a powerful classifier cannot be tested on all image sub-windows in a reasonable amount of time. Thus we propose a novel three-stage classifier, which combines linear, quasi-linear, and non-linear kernel SVMs. We show that increasing the non-linearity of the kernels increases their discriminative power, at the cost of an increased computational complexity. Our contributions include (i) showing that a linear classifier can be evaluated with a complexity proportional to the number of sub-windows (independent of the sub-window area and descriptor dimension); (ii) a comparison of three efficient methods of proposing candidate regions (including the jumping window classifier of Chum and Zisserman (CVPR 2007) based on proposing windows from scale invariant features); and (iii) introducing overlap-recall curves as a mean to compare and optimize the performance of the intermediate pipeline stages.

The method is evaluated on the PASCAL Visual Object Detection Challenge, and exceeds the performances of previously published methods for most of the classes.

1. Introduction

Our objective in this paper is object category *detection*: the task of determining if one or more instances of a category are present in an image and, if they are, localize all instances by specifying tight bounding boxes around them. The simpler task of image *classification*, where all that must be determined is if there is a class instance in the image (but not localization), has enjoyed quite some success recently. For the Caltech 101/256 databases [12] high performance has been achieved by (i) using multiple feature types [25]; (ii) including spatial information [14]; and (iii) optimizing the combination of features and spatial pyramid levels using multiple kernel learning [21] for an SVM classifier. These state-of-the-art performances are achieved for images that for the most part are well aligned, where the object dominates the image and where there is little background clutter.

The questions we investigate here[1] are: (i) can these methods for image classification be successfully applied to detection? i.e. to localize the object under much more challenging conditions (pose variation, background clutter); and (ii) if they can be used as a detector, how does their performance compare to the state of the art? As a test bed we use the PASCAL Visual Object Classes (VOC) training and testing data, since state of the art detectors are annually compared on these images [9].

A natural starting point is to apply the state-of-the-art image classifier of [21] as a sliding window detector, in the manner of Viola and Jones [24]. However, a naive implementation is computationally infeasible because: (i) regions must be searched over position, scale, and aspect ratio, (ii) each region is described by high dimensional feature histograms (due to the combination of multiple feature channels and spatial subdivisions), and (iii) the classifier uses non-linear kernels and thousands of support vectors.

While methods to accelerate the relevant computations have been proposed in [13, 16], we show in Sec. 3 that they do not achieve a sufficient speed-up if the most powerful model is to be used. As in [24], we therefore adopt a cascade approach, and to this end, we introduce a novel multi-stage classifier, where each stage employs a more powerful (and expensive) classifier that is a closer approximation to our goal. In Sec. 3.1 we compare three very fast classifiers that are suitable for the first stage of the cascade. The first two are linear classifiers (again consisting of multiple kernels) that we compute extremely efficiently from a single

[1]We regret any confusion caused by the BMVC 08 paper by Bosch, Gulshan, Varma and Zisserman which we withdrew. The algorithms and results here should be taken as a replacement for the BMVC 08 paper.

2009 IEEE 12th International Conference on Computer Vision (ICCV)
978-1-4244-4419-9/09/$25.00 ©2009 IEEE

integral image. This saving was not employed in previous work and is essential if high dimensional feature histograms are used. The first classifier uses fixed aspect windows as is very commonly done [1, 7, 10], and the second considers multiple aspect ratios learned from the data. The third classifier is a 'jumping window' [6] which, again, is able to model the aspect ratio variation in the training set. The output of these classifiers is a set of candidate regions, about 2000, for each image, that are then passed onto the more powerful classifiers of the later stages of the cascade. To compare the performance of the three types we introduce a *recall-overlap* performance curve.

In Sec. 3.2 we discuss the second and third stages, which are based on the more expensive quasi-linear χ^2 and non-linear RBF-χ^2 kernels respectively. While evaluating the quasi-linear classifier is independent of the number of support vectors, the necessity of computing a large number of high dimensional histograms for each candidate region makes this classifier about a thousand times slower than the classifiers used in the first stage. Finally, we pass to the last stage around 100 candidates. The last stage is overall the slowest, even for the very small number of candidates on which it is evaluated, due to both the large number of support vectors and the high dimensionality of the histograms. In Sec. 3.3 we discuss how the feature histogram normalization affects both efficiency and modeling quality.

We describe the features and implementation details in Sec. 4.1, the learning strategy in Sec. 4.2, and evaluate the performance in Sec. 5. We apply this approach to the PASCAL VOC challenge and show that indeed, the MKL image classifier can be ported to a detector, and that this approach surpasses in most cases all other methods todate that have reported results on this dataset.

Related work. The method most similar to ours is the INRIA PlusClass entered in the PASCAL VOC Challenge [9]. While both our and their method share the use of multiple features and a cascade, their emphasis is on the use of contextual information, while ours is on improving the quality and efficiency of the object representation. So for instance they use a cascade of two rather than three stages, and two feature channels rather than six (and eighteen kernels) as we do. While exploiting context is an interesting route to explore, currently our method outperforms theirs in all but a few categories without using contextual information.

Other relevant work will be discussed where appropriate throughout the text.

1.1. The PASCAL VOC Detection Challenge

The PASCAL Visual Object Detection Challenge (VOC) [9] data consists of a few thousand images annotated with bounding boxes for objects of twenty categories (e.g., car, bus, airplane, ...). The *detection* challenge is the following: predict the bounding box and label of each object from the target classes in a test image. Each bounding box is output together with a confidence value, and this value is used to generate a precision-recall graph for each class. Detections are considered true or false positives based on their *overlap* with ground truth bounding boxes. The overlap between a proposed bounding box R and the ground-truth box Q is computed as

$$\frac{\text{area}|Q \cap R|}{\text{area}|Q \cup R|}. \tag{1}$$

An overlap of 50% or greater is labeled as true positive. Any additional overlapping bounding box (duplicate detections) are rejected as false positives. Performance is then measured by the *average precision* (AP). Full details of the challenge, including the results of all participants, are given at [9].

2. Multiple kernel sliding-window classifier

Our aim is to learn an SVM classifier [19] where, rather than using a pre-specified kernel, the kernel is learnt to be a linear combination of given base kernels [2, 21]. The classifier defines a discriminant function $C(\mathbf{h}^R)$ that is used to rank candidate regions R by the likelihood of containing an instance of the object of interest. The classifier argument $\mathbf{h}^R$ is a collection of feature histograms $\mathbf{h}^R = \{h^R_{fl}\}$, for multiple feature channels f and spatial pyramid levels l (Sec. 4.1). Captial letters H^R_{fl} will denote the *unnormalized* histograms, i.e. the raw feature count. Note that $h^R_{fl} = H^R_{fl}/\|H^R_{fl}\|$ where $\|\cdot\|$ denotes l^1 or another appropriate norm.

The function $C(\mathbf{h}^R)$ is learnt, along with the optimal combination of features and spatial pyramid levels, by using the Multiple Kernel Learning (MKL) technique proposed in [21]. The function $C(\mathbf{h}^R)$ is the discriminant function of a Support Vector Machine (SVM), and is expressed as

$$C(\mathbf{h}^R) = \sum_{i=1}^{M} y_i \alpha_i K(\mathbf{h}^R, \mathbf{h}^i). \tag{2}$$

Here $\mathbf{h}^i$, $i = 1, \ldots, M$ denote the descriptors of M training regions, selected as representative by the SVM, $y_i \in \{+1, -1\}$ their class labels, and K is a positive definite (PD) kernel, obtained as a liner combination of histogram kernels

$$K(\mathbf{h}^R, \mathbf{h}^i) = \sum_{fl} d_{fl} K(h^R_{fl}, h^i_{fl}). \tag{3}$$

MKL learns both the coefficient α_i and the histogram combination weights $d_{fl} \geq 0$. Following the method of [4], a different set of weights $\{d_{fl}\}$ are learnt *for each class* as detailed in Sec. 4.2. Weights can therefore emphasise more discriminative features for a class or pyramid level, and even ignore features/levels that are not discriminative by setting d_{fl} to zero.

Because of linearity, (2) can be rewritten as $C(\mathbf{h}^R) = \sum_{fl} d_{fl} C(h^R_{fl})$, where

$$C(h^R) = \sum_{i=1}^{M} y_i \alpha_i K(h^R, h^i) \tag{4}$$

for each histogram $h^R = h^R_{fl}$ and $h^i = h^i_{fl}$.

Choice of elementary kernels $K(h, h^i)$. We consider three types of kernels, differing in their discriminative power and computational cost. Our gold standard, and most expensive, classifier [21] uses non-linear RBF-χ^2 kernels of the form

$$K(h, h^i) = e^{-\gamma \chi^2(h, h^i)}. \tag{5}$$

We also consider "quasi-linear" kernels of the form

$$K(h, h^i) = \frac{1}{2}(1 - \chi^2(h, h^i)) \tag{6}$$

and linear kernels of the form

$$K(h, h^i) = \langle h, h^i \rangle. \tag{7}$$

In total six features are used (including bag-of-words, dense visual words, self-similarity descriptors, and edge based descriptors) and three pyramid levels for each (corresponding to one, four, and sixteen spatial subdivisions), with a kernel corresponding to each feature (Sec. 4.1). Therefore 18 weights d_{fl} need to be learned for their linear combination.

3. Inference cost and cascade

The main technical obstacle is searching for the best matching region R^*, i.e. solving the *inference problem* $R^* = \operatorname{argmax}_R C(\mathbf{h}^R)$. This applies to learning as well, as this is done by bootstrapping and entails performing inference multiple times (Sec. 4.2).

Exhaustive search requires a number of operations proportional to the number N of regions R to be tested, the dimensionality B of the histograms, and the number M of support vectors in the calculation of $C(\mathbf{h}^R)$, i.e. the complexity is $O(BNM)$. As will be seen, this complexity is prohibitively expensive as in our case $N \approx 10^5$ (because we search over translation, scale, and aspect ratio), $B \approx 10^4$ (because we combine several high dimensional histograms) and $M \approx 10^3$. In the following we first introduce a taxonomy of kernels, and then discuss for each type of kernel if, and how, this cost can be reduced. In particular we show that for a linear kernel the complexity can be reduced to $O(N)$.

Kernel taxonomy. All popular histogram kernels are of one of the following types: *linear*, *quasi-linear* (e.g. χ^2, intersection, Hellinger's), and *non-linear* (e.g. RBF). This taxonomy can be characterized as follows. Let b be the histogram bin index; all such kernels may be written as

$$K(h, h') = f\left(\sum_{b=1}^{B} g(h_b, h'_b)\right) \tag{8}$$

for a choice of the functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R}^2 \to \mathbb{R}$. For linear kernels, both f and g are linear functions, and for a quasi-linear kernel only f is linear:

type	example	$f(z)$	$g(x,y)$	eval. complexity
linear	linear	z	xy	$O(N)$
quasi-lin.	χ^2	z	$\frac{2xy}{(x+y)}$	$O(BN)$
non-lin.	RBF-χ^2	$e^{-\gamma z}$	$\frac{(x-y)^2}{(x+y)}$	$O(MBN)$

SVM evaluation cost. The major bottleneck in performing inference is the calculation of the histograms $\mathbf{h}^R$ for all the N regions R for which the classifier $C(\mathbf{h}^R)$ must be evaluated. Denote by H^p the feature count for pixel p; then the un-normalized histogram H^R for the region R (i.e. the raw feature count) can be written as

$$H^R = \sum_{p \in R} H^p. \tag{9}$$

This calculation requires at least $\Omega(BN)$ operations as all bins must be visited at least once. In our case $BN \approx 10^9$, so just *computing all the SVM inputs* $\mathbf{h}^R$ *is prohibitively slow*.

Cascade. Our solution is to use a cascade of increasingly powerful classifiers. As seen, it is crucial that the first stage avoids computing explicitly all the histograms $\mathbf{h}^R$. Therefore we introduce a technique to evaluate $C(\mathbf{h}^R)$ without computing $\mathbf{h}^R$ if a linear kernel is used. The intermediate stage uses a better model (quasi linear kernel) for which a known speed-up can be applied [16]. The last stage uses the most powerful and expensive model (non-linear kernel), but it is evaluated on a small number of candidate regions which have been filtered by previous cheaper stages.

3.1. First stage: Fast SVM vs jumping window

The first stage of the classifier proposes candidate regions which are then classified by the more powerful, but slower, later stage classifiers. An ideal first stage of a cascade should: (i) reject all regions that do not contain an object instance; (ii) keep all regions that do contain an object instance; and (iii) do this at low cost. Of course, such an ideal first stage would then also be an ideal detector, and the subsequent stages would not be required. In practice, there is a trade-off between (i) and (ii): we seek an operating point that rejects as few true regions as possible (high recall), whilst rejecting as many false regions as possible (high precision). We introduce below a curve for quantifying this trade-off. First, we describe and compare the three types of classifiers suitable for the first stage of the cascade.

Fast linear SVM. For a linear SVM and an un-normalized histogram

$$C(H^R) = \sum_{i=1}^{M} y_i \alpha_i \langle H^i, H^R \rangle \tag{10}$$

and, as is well known for this case, the sum over support vectors can be precomputed as $\mathbf{w} = \sum_{i=1}^{M} y_i \alpha_i H^i$, so that evaluating $C(H^R) = \langle \mathbf{w}, H^R \rangle$ is independent of M. However, the evaluation cost is still $O(BN)$ and we show here that, for un-normalized or l^1-normalized histograms, the linear SVM can be evaluated in time which is proportional to the number N of regions only. The reason is that the bin dimension b can be projected on the linear SVM weight vector *before* the discriminant scores $C(H^R)$ are evaluated, thus removing the factor B from the complexity. Using (9) and $C(H^R) = \langle \mathbf{w}, H^R \rangle$ one obtains

$$\begin{aligned} C(H^R) &= \sum_{b=1}^{B} \mathbf{w}_b H_b^R = \sum_{b=1}^{B} \mathbf{w}_b \sum_{p \in R} H_b^p \\ &= \sum_{p \in R} \left(\sum_{b=1}^{B} \mathbf{w}_b H_b^p \right) = \sum_{p \in R} \psi(p) \end{aligned}$$

where $\psi(p) = \sum_{b=1}^{B} \mathbf{w}_b H_b^p$, i.e. the summations over p can be moved outside the sum over b. Thus evaluating the un-normalized linear SVM $C(H^R)$ reduces to first calculating the integral image of the score map $\psi(p)$ (in $O(P)$ time if at most a constant number of features occur for each pixel) and then computing the score for each of the N regions R in time $O(N)$. To calculate $C(h^R)$ for an l^1-normalized histogram it is also necessary to repeat the process for the mass map $M(p) = \sum_{b=1}^{B} H_b^p$ in order to normalize the scores, but the complexity is the same. Thus, evaluating the linear SVM requires $O(N + P)$ operations only, which is independent of both the number of histogram bins B and support vectors M. Consequently, evaluating the linear SVM is feasible even for a number of regions as large as $N \approx 10^5$.

However, as will be discussed in Sec. 3.3, a bias is introduced by not l^2-normalizing the histograms, which favors either small (in case of l^1 normalization) or large (in case of no normalization) regions. Because of this bias, regions can be reliably compared only if they have roughly the same size. Thus candidate regions are grouped by size and an equal number of highly-ranked regions from each group are extracted.

The SVM classifiers slides 100–200 representative bounding boxes that span the characteristic scales and aspect ratios of the object category. Each box is translated by steps of 10% its size to cover the image support.

The representative boxes are obtained as centers of clusters of training bounding boxes constructed by running agglomerative clustering based on the overlap measure (1). We consider also a fixed-aspect variation of the SVM classifier for which the cluster centers may vary in scale but are constrained to have the same aspect ratio.

Jumping window. This is the approach of Chum and Zisserman [6]. Discriminative visual words [5] are learnt for the object class (from the training and a negative set), and their distribution in each image determines the candidate proposals. Discriminability is measured using the likelihood ratio discriminability function D of [8],

$$D(w) \sim \frac{\#\text{target object instances containing } w}{\#\text{object instances containing } w}.$$

and provides a ranking of the visual words.

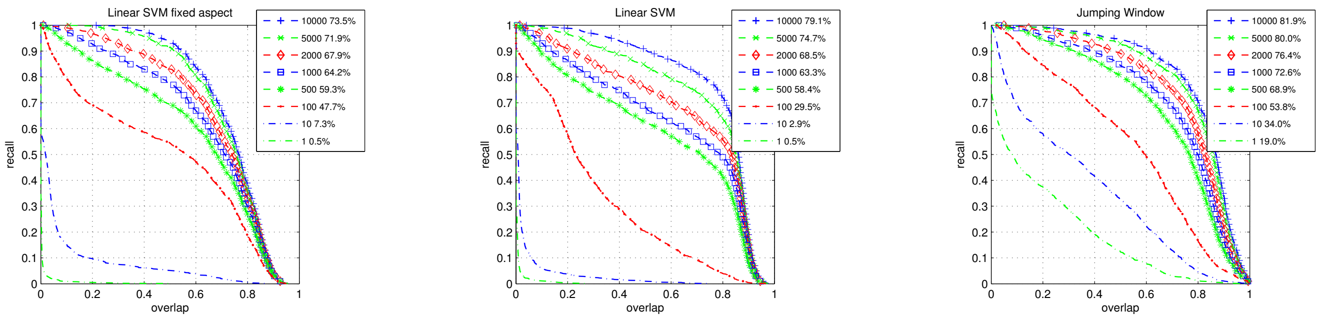
Individual discriminative visual words are used to generate a hypotheses for the class instance location. In detail, a hypothesis is a pair (w, R) of visual word w and a rectangular region R. The rectangle represents the regions with fixed relative position and scale with respect to the position and scale of the visual word w. The pairs (w, R) are learnt from the provided regions of interest in the training images. In the training images there will be a number of rectangles R_i associated with w; similarly to [17], these are aggregated into a single rectangle using mean-shift clustering (using [23] for speed).

Comparison. Evaluation of both the linear SVMs is fast, requiring overall only two to three seconds per image, but the jumping window is even faster (under a second for each image). In order to compare the quality of the candidates produced by the three methods, and to establish a reasonable trade-off between speed (of the next cascade stages), precision and recall, we introduce the *recall-overlap* curves. Given a number of candidate regions per image, the recall-overlap curve is obtained by measuring the recall rate of the ground-truth object bounding boxes for a given minimum overlap (1).

These curves can be used to compare the recall rates of the three classifiers at any level of overlap. Assuming that the later stages of the classifier require sufficient overlap of the candidates with the ground truth, we choose 80% overlap as our operating point. It is clear from Fig. 1 that the jumping window performs best at this point – and it is also faster than the sliding window linear classifier, so it is the clear choice for the first stage. 2000 candidates are selected as this gives a 70% recall rate, and this is an upper bound on the recall for the next stages.

3.2. Second and third stages

Our second and third stages use respectively a “quasi-linear” (reducing the candidates to 100) and non-linear ker-



(a) (b) (c)

Figure 1. **Recall-Overlap curves for the three first stage classifiers.** The quality of the candidates generated by three different methods is compared: (a) linear SVM sliding windows of multiple scales but single aspect ratio; (b) linear SVM sliding windows of multiple scales and aspect ratios; (c) jumping window (also using multiple scales and aspect ratios). This example is from the class *car*, but other classes are qualitatively similar. The figure shows overlap-recall curves for 1, 10, 100, 500, 2,000, 5,000 and 10,000 candidates and reports the AUC for each curve. 2,000 of such candidates are then filtered by two additional pipeline stages. Thus the goal is to obtain high recall and overlap for the 2,000 candidate curve. For instance, at 80% minimum overlap, the linear SVM with fixed aspect ratio achieve only 30% recall. Optimizing over aspect ratios improves this result to about 50% and 60% for respectively the linear SVM and the jumping window. Notice also that the jumping window performs better in most cases (and is much faster than the linear SVM).

nels (yielding the final scores, to which non-maxima suppression is applied as explained in Sect. 4.2). For general non linear kernels, the evaluation of $C(h^R)$ requires comparing the histogram h^R of each region R to each of the M support vector histograms h^i. For B bins and N regions, this requires $O(BNM)$ operations. For quasi-linear kernels [16] show that (2) can be approximated in $O(BN)$ time (i.e. independent of the number M of support vectors). This is done by rewriting (4) as

$$C(h) = \sum_{i=1}^{M} y_i \alpha_i \sum_{b=1}^{B} g(h_b, h_b^i) = \sum_{b=1}^{B} \psi_b(h_b) \qquad (11)$$

where $\psi_b(z) = \sum_{i=1}^{M} y_i \alpha_i g(z, h_b^i)$, and fitting a piecewise-linear approximation to the functions $\psi_b : \mathbb{R} \to \mathbb{R}$, $b = 1, \ldots, B$ (for the intersection or l^1 kernel this approximation can be exact).

Lampert *et al.* [13] propose branch-and-bound to accelerate inference by reducing the number N of regions visited, but demonstrate the algorithm for un-normalized histograms and a linear SVM. In an attempt to bypass the cascade and to use directly the quasi-linear or non-linear models (as the linear model is suboptimal, see [7, 16] and Sec. 3.2), we derived bounds to apply [13] to these cases and hence obtained a reduction of the factor N comparable to [13]. Unfortunately, we still found it necessary to visit several thousand regions per image, which, combined with the large histogram dimension, makes this approach inpractical. This motivates the use of a cascade of classifiers.

3.3. Histogram normalization and bias

Histograms are used here as *descriptors of the region appearance*. In particular, the kernel $K(h^R, h^{R'})$ is intended as measure of similarity of the appearance of the regions R and R', and should attain a maximum when the two regions have identical appearance, i.e. the requirement that $K(h^R, h^R) \geq K(h^R, h^{R'})$ for all regions R and R'. To find a sufficient condition for this requirement, note that any PD kernel $K(h^R, h^{R'})$ can be turned into a square distance by the formula $d^2(h^R, h^{R'}) = K(h^R, h^R) + K(h^{R'}, h^{R'}) - 2K(h^R, h^{R'})$, and that $0 = d^2(h^R, h^R) \leq d^2(h^R, h^{R'})$ by the axioms of distance. Hence a sufficient condition, as can be verified by substitution in the previous inequality, is that $K(h, h) =$ const. for all histograms h. For the linear kernel, $K(h, h) = \|h\|_2^2$ and the condition is satisfied *if the histograms are l^2 normalized*. The linear SVMs with l^1-normalized or un-normalized histograms fail to meet this condition but are extremely efficient to evaluate (Sect. 3.1). The questions is, does this adversely affect the classification?

Consider first l^1-normalized histograms. The discriminant score can be calculated and bounded as $C(h^R) = \sum_b \mathbf{w}_b h_b^R \leq (\max_b \mathbf{w}_b) \sum_b h_b^R = \max_b \mathbf{w}_b$ because, by hypothesis, $\|h^R\|_1 = \sum_b h_b^R = 1$. In particular, the upper bound is attained if the mass of h^R is concentrated on the bin b with the largest weight $\mathbf{w}_b$. This happens, for instance, if R is a small region that encloses just a single occurrence of a feature of that label. The consequence is that smaller regions are likely to have a larger score magnitude than than larger regions.

Consider now using un-normalized histograms H^R and a

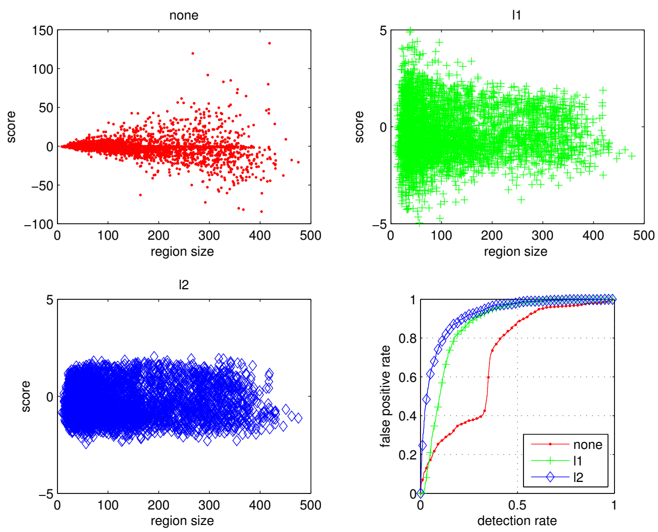


Figure 2. **Normalization and bias.** A linear SVM is trained to discriminate image regions R that do and do not portray a car (for VOC 07 training and testing). The figure compares using un-normalized, l^1-normalized, and l^2-normalized histograms. The first three panels are scatter plots of the SVM score $C(h^R)$ (for some positive and negative test regions) and the square root of the region area. The un-normalized scores reward large regions, the l^1-normalized scores small regions, and the l^2-normalized scores are essentially unbiased. The last panel shows the ROC curve for the three classifiers, illustrating that l^2 normalization performs best.

linear kernel as in [13]. In this case, the score of a region R is given by $C(H^R) = \langle \mathbf{w}, H^R \rangle = \|H^R\|_1 \langle \mathbf{w}, \frac{H^R}{\|H^R\|_1} \rangle$ and is proportional to the mass (number of feature occurrences) $\|H^R\|_1$, which is usually strongly correlated with the region *area*. The consequence is that larger regions are likely to have a larger score magnitude than than smaller regions.

Fig. 2 illustrates such considerations empirically. Unfortunately, we are not aware of any method that could be used to evaluate the l^2-normalized SVM in $O(N)$ operations. Thus the fast linear SVM should be considered a weak classifier suitable for the first cascade stage only. Note that for the quasi-linear and non-linear kernels normalization is not a bottleneck and the proper one can be used (which, based on the proposed criterion, is l^1).

4. Features and implementation details

4.1. Appearance descriptors

The descriptors of the appearance of the candidate regions R are constructed from a number of different feature channels. These are the features used in [4, 13, 21, 25], and are computed using open source code [22].

Bag of visual words. SIFT descriptors [15] are extracted at Hessian-Laplace [18] points and quantized in a vocabulary of 3000 words, trained on features from the bounding boxes of several object instances. The vocabulary is then discriminatively compressed down to 64 words for each class by using [11].

Dense words (PhowGray, PhowColor) [3]. Rotationally invariant SIFT descriptors are extracted on a regular grid each five pixels, at four multiple scales (10, 15, 20, 25 pixel radii), zeroing the low contrast ones. Descriptors are then quantized in 300 visual words. The color versions stacks SIFT descriptors for each HSV color channels.

Histogram of oriented edges (Phog180, Phog360) [7, 3]. The Canny edge detector is used to compute an edge map and the underlying image gradient $\nabla I(p)$ is used to assign an orientation and a weight to each edge pixel p. The orientation angle is then quantized in eight bins with soft linear assignment and an histogram is computed.

Self-similarity features (SSIM). Self-similarity descriptors [20] are computed on a regular grid at steps of five pixels. Each descriptor is obtained by computing the correlation map of a 5×5 pixels patch in a window of radius 40 pixels, then quantizing it in 3 radial bins and 10 angular bins, obtaining 30 dimensional descriptor vectors. The descriptors are then quantized into 300 visual words.

Spatial pyramid. For each feature channel a three-level pyramid of spatial histograms h^R_{f0}, h^R_{f1}, h^R_{f2} is comptued, similar to [6, 14].

4.2. Learning the object classifier

Each SVM classifier $C(h^R)$ is trained to discriminate between candidate regions R that do and do not contain an instance of the object of interest, i.e. a one-vs-the-rest classifier.

Training (performed using the MKL algorithm from [21]) requires providing a number of positive and negative data samples. The ground truth object instances for a class, plus a number of jittered instances (obtained by flipping the training images), are used as positive samples.

Regions that do not overlap the target object instances by more than 20% are used as negative samples. The number of possible negative regions is prohibitively large and a set of representative cases must be identified. This is done by *retraining* (bootstrapping) each classifier, as follows. The classifier is used to extract candidate regions from a number of training images. The candidates are then compared to the ground truth, and are labeled as errors if they overlap the target class by less than 20%. Finally, up to three highly scored errors are extracted for each image, avoiding mutual overlap of more than 50%. Such errors are then added to the SVM training data as hard negative samples and the SVM

	aerop.	bicyc.	bird	boat	bottle	bus	car	cat	chair	cow	dinin.	dog	horse	motor.	person	potte.	sheep	sofa	train	tvmon.
st2	**37.6**	**47.8**	**15.3**	**15.3**	21.9	**50.7**	**50.6**	**30.0**	**17.3**	**33.0**	**22.5**	**21.5**	**51.2**	45.5	23.3	12.4	23.9	**28.5**	**45.3**	**48.5**
st1	36.6	42.5	12.8	14.5	15.1	46.4	45.9	25.5	14.4	30.4	19.0	16.0	49.0	**46.0**	21.5	11.0	**24.5**	26.4	42.6	40.8
dt	18.0	41.1	9.2	9.8	**24.9**	34.9	39.6	11.0	15.5	16.5	11.0	6.2	30.1	33.7	**26.7**	**14.0**	14.1	15.6	20.6	33.6
v7	26.2	40.9	9.8	9.4	21.4	39.3	43.2	24.0	12.8	14.0	9.8	16.2	33.5	37.5	22.1	12.0	17.5	14.7	33.4	28.9

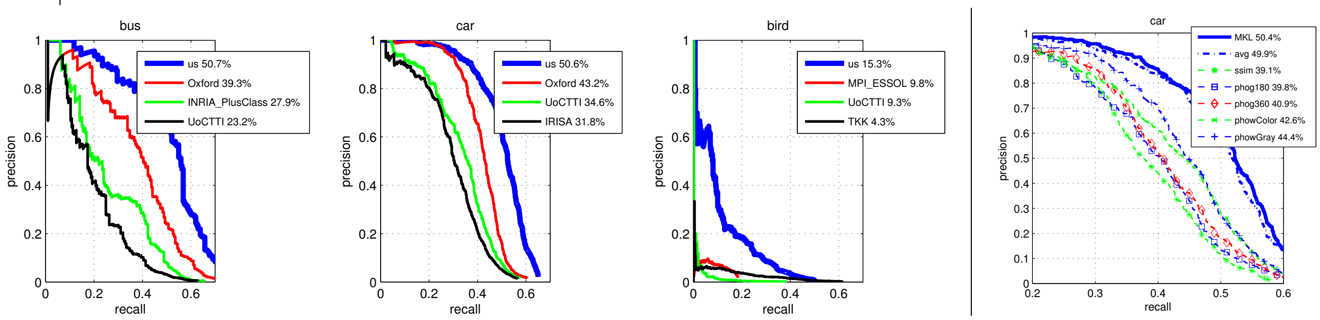


Figure 3. **Training and testing on VOC 2007.** The table reports the average precision obtained by our method in each of the 20 PASCAL 2007 challenge categories. The method has been trained and tested on the 2007 data. (st2) refers to stage 2 of the pipeline (non-linear SVM) and (st1) to stage 1 (quasi-linear SVM). For comparison, (dt) reports the results from [10] and (v7) the best result for each category among all methods submitted to the VOC 2007 challenge (see [9] for the breakdown). Our method outperforms the others in all but three categories. With few exceptions, the non-linear SVM (st2) outperforms the quasi-linear SVM (st1). Below are the precision-recall curves obtained for the two well performing classes and one difficult one. **Last panel.** The following discriminative models are compared for the class *car*: MKL, avg (average of all channels), ssim (self-similarity), phog180, phog360, phowColor, phowGray. Combining features yields a large improvement; averaging is close to MKL, but the latter yields a sparse selection of channels.

	aerop.	bicyc.	bird	boat	bottle	bus	car	cat	chair	cow	dinin.	dog	horse	motor.	person	potte.	sheep	sofa	train	tvmon.
st2	**41.2**	39.0	**16.8**	**19.1**	24.8	28.4	**37.3**	**29.1**	13.8	16.7	12.3	**17.9**	**39.8**	**43.7**	25.0	7.9	**19.4**	**17.4**	**36.5**	**41.2**
st1	40.6	34.1	13.6	13.7	17.4	**28.6**	33.3	21.4	12.1	15.1	**15.1**	15.0	37.9	39.5	24.0	10.2	18.6	15.1	35.4	39.3
v8	36.5	**42.0**	11.3	11.4	**28.2**	23.8	36.6	21.3	**14.6**	**17.7**	**15.1**	14.9	36.1	40.3	**42.0**	**12.6**	**19.4**	17.3	29.6	37.1

Figure 4. **Training and testing on VOC 2008.** (st2) non-linear classifier (st1) quasi-linear classifier (v8) best result for each category among all methods submitted to the VOC 2008 challenge (see [9] for the breakdown).

	aerop.	bicyc.	bird	boat	bottle	bus	car	cat	chair	cow	dinin.	dog	horse	motor.	person	potte.	sheep	sofa	train	tvmon.
st2	**38.3**	**41.3**	**15.5**	**14.6**	17.6	**45.4**	**49.8**	**25.6**	15.2	23.6	7.7	**18.4**	**40.7**	**43.8**	**21.3**	**10.6**	19.4	18.6	**42.3**	**45.1**
st1	37.2	38.5	12.9	6.0	14.9	43.8	45.5	17.9	12.9	21.4	**10.6**	16.4	37.1	40.8	20.5	6.2	19.3	16.1	38.6	40.6
v8	28.5	39.0	10.7	11.2	**20.2**	41.0	48.4	15.2	**16.1**	**25.7**	10.1	11.5	34.9	39.7	16.8	10.3	**21.8**	**22.8**	37.0	36.3

Figure 5. **Training on VOC 2008 and testing on VOC 2007.** (st2) non-linear classifier (st1) quasi-linear classifier (v8) best result for each category among all methods submitted to the VOC 2008 challenge (see [9] for the breakdown).

is trained again.

Since extracting candidates is a relatively slow operation, retraining is operated on rotating subsets of training images as follows. Training images are partitioned into two subsets, making sure that each subset contains in roughly equal proportions images with the target object (e.g. car), other easily confused objects (e.g. bus), and other objects as well. The model is then tested on each subset in turn, including new hard negative and retraining each time. Experimentally, we verified that it is beneficial to retrain twice on each subset.

Post-processing. The output of the last stage is a ranked list of 100 candidate regions per image. Many of these regions correspond to duplicate detections, which we remove in post processing, by non-maxima suppression. This is implemented as follows: The most highly ranked candidate is selected, all other candidates with an overlap greater than 20% are removed and the process is repeated until at most ten candidates are selected (as images typically do not contain more than a few instances of an object).

5. Experiments

We evaluated our method on the VOC 2007 (Fig. 3) and 2008 (Fig. 4 and 5) challenges, outperforming all other methods in all but a few cases. Since the model is appearance and shape based, it works well for categories which are characterized by such properties (e.g. the vehicle classes and some animal classes like horse). Performance is less good for some wiry objects (e.g. potted plant) or objects defined more by their function and context than by their appearance (e.g. chair, dining table).

Effect of combining features. The last panel of Fig. 3 illustrates the effect of combining feature channels on one class (other classes are qualitatively similar). While combining features is very beneficial, the gain obtained by MKL over simple averaging is modest. However MKL determines a sparse selection of features, which helps improving the efficiency of inference.

Simplifying the training data. We found it beneficial to remove from the positive training data truncated objects (based on the ground truth annotations). Our interpretation is that training on partial detections makes learning more ambiguous and difficult.

Testing times. On a standard 3GHz CPU the following testing times per image were observed: below 1 second for the jumping window stage, around 2–3 seconds for the linear SVM stage (on 2×10^5 sliding windows), around 15 seconds for the quasi-linear kernel stage (on 2×10^3 candidates), around 50 seconds for the non linear kernel stage (on 100 candidates). Running the last stage directly on the sliding windows would require 27 hours per image (as opposed to roughly 67 seconds taken by our method).

6. Discussion

We have shown that a MKL classifier can be successfully trained and tested, in a reasonable time, to act as a detector. The performance exceeds in almost all cases the state of the art on the PASCAL VOC benchmark. This answers the questions raised in the introduction.

However, we chose a three stage cascade to overcome the complexity cost, and this has resulted in quite a 'heavy' algorithm in both training and testing. We are currently investigating alternatives to the full cascade, for example using the candidates from the first stage as regions to search around, rather than the only possibilities for further consideration. In this way we hope to reduce the number of candidates required, and also the power required by the subsequent classifiers.

Acknowledgments. We are grateful for funding from the EU under PASCAL2, CLASS and ERC VisRec no. 228180; and the RAEng, Microsoft, and ONR MURI N00014-07-1-0182.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE PAMI*, 20(11):1475–1490, 2004.

[2] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proc. ICML*, 2004.

[3] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification via pLSA. In *Proc. ECCV*, 2006.

[4] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proc. CIVR*, 2007.

[5] C. Bouveyron, S. Girard, and C. Schmid. Class-specific subsace discriminant analysis for high-dimensional data. *Lecture Notes in Computer Science*, 3940, 2006.

[6] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proc. CVPR*, 2007.

[7] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. In *Proc. CVPR*, volume 2, pages 886–893, 2005.

[8] G. Dorkó and C. Schmid. Object class recognition using discriminative local features. *IEEE PAMI*, 2004.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. `http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html`, 2008.

[10] P. F. Felzenszwalb, D. McAllister, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. CVPR*, 2008.

[11] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *Proc. ECCV*, 2008.

[12] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[13] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localizationby efficient subwindow search. In *Proc. CVPR*, 2008.

[14] S. Lazebnik, C. Schmid, and J. Ponce. Beyound Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proc. CVPR*, 2006.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[16] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Proc. CVPR*, 2008.

[17] M. Marszałek and C. Schmid. Spatial weighting for bag-of-features. In *Proc. CVPR*, 2006.

[18] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. ICCV*, 2001.

[19] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[20] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Proc. CVPR*, 2007.

[21] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proc. ICCV*, 2007.

[22] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[23] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Proc. ECCV*, 2008.

[24] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, pages 511–518, 2001.

[25] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2007.