

# Support Kernel Machines for Object Recognition

Ankita Kumar  
University of Pennsylvania

Cristian Sminchisescu  
TTI-Chicago

## Abstract

*Kernel classifiers based on Support Vector Machines (SVM) have recently achieved state-of-the-art results on several popular datasets like Caltech or Pascal. This was possible by combining the advantages of SVM – convexity and the availability of efficient optimizers, with ‘hyperkernels’ – linear combinations of kernels computed at multiple levels of image encoding. The use of hyperkernels faces the challenge of choosing the kernel weights, the use of possibly irrelevant, poorly performing kernels, and an increased number of parameters that can lead to overfitting. In this paper we advocate the transition from SVMs to Support Kernel Machines (SKM) – models that estimate both the parameters of a sparse linear combination of kernels, and the parameters of a discriminative classifier. We exploit recent kernel learning techniques, not previously used in computer vision, that show how learning SKMs can be formulated as a convex optimization problem, which can be solved efficiently using Sequential Minimal Optimization. We study kernel learning for several multi-level image encodings for supervised object recognition and report competitive results on several datasets, including INRIA pedestrian, Caltech 101 and the newly created Caltech 256.*

## 1. Introduction

Recent work in object recognition and image classification has shown that significant performance gains can be achieved by carefully combining multi-level, coarse-to-fine, layered feature encodings, and learning methods. Top-scoring classifiers on image databases like Caltech or Pascal tend to be discriminative and kernel-based [19, 13, 7, 5], but generative methods [6, 18] can also be used in order to build hybrid, even more sophisticated kernels (*e.g.* the Fisher kernel) [9]. Monolithic kernels are by no means the only way to build successful classifiers. Several hierarchical methods like HMAX [17, 14] or recent versions of convolutional neural networks [16] use them more sparingly, at a final stage of a complex hierarchical computation that involves successive convolution and rectification, but a straightforward monolithic kernel can be alternatively ob-

tained by combining representations across all layers. In any case, an underlying theme of current research is the use of kernel methods, in particular the Support Vector Machine (SVM) – a methodology well-justified both theoretically and practically [4]: the resulting program is convex with global optimality guarantees, and efficient algorithms like Sequential Minimal Optimization (SMO) can be used to solve large problems, with hundreds of thousands of examples.

The successful proliferation of a variety of kernels has recently motivated several researchers to explore the use of homogenous models obtained as linear combinations of histogram intersection kernels, computed at multiple levels of image encoding [7, 13]. These more sophisticated classifiers have been demonstrated convincingly and state-of-the-art results have been achieved, but their use raises a number of new research challenges:

(i) A *weighting of kernels* needs to be specified. In particular, for histogram intersection, promising results have been obtained using geometric approximate weightings to the optimal bipartite matching<sup>1</sup> – but this problem-dependent intuition may not be available for all kernels, or may not be as effective when combining heterogeneous kernels with different feature spaces, *e.g.* histogram intersections, polynomials and RBFs.

(ii) The *kernel selection* becomes important especially for small training sets, as the effective number of model parameters increases with the number of kernels. This raises the question of which kernel is good, which one doesn't matter and which one is likely to harm performance, a problem that requires capacity control. Indeed, previous studies [13, 3] have shown that for some problems, the best performance is achieved only for a subset of the kernels / levels of encoding – sometimes only a single one. The insights were gained by learning classifiers both for individual kernels and for multiple ones, but as their number increases, an exhaustive exploration of the kernel power set becomes infeasible. One is faced with the combinatorial problem of selecting a parsimonious kernel subset for a desired accuracy-computation trade-off – this is precisely the *sparse, kernel*

<sup>1</sup>These penalize matchings found in large histogram cells more than ones found in fine grained cells / layers.

*subset selection problem* we consider.

The main emphasis of our work is to advocate the transition from Support Vector Machines – convex learners that can be used to train one kernel efficiently using SMO, to Support Kernel Machines (SKM) – convex and sparse learners that can be used to *train multiple kernels* by jointly optimizing both the coefficients of a conic combination of kernel matrices and the coefficients of a discriminative classifier. The research builds on recent algorithms [12, 1, 2], where the objective of learning a sparse combination of kernels is formulated as a convex, quadratically constrained quadratic problem, efficiently solved using sequential minimal optimization (SMO) techniques.

In independent work, [3, 11] have also studied the problem of learning a combination of kernels. De la Torre *et al* [11] learn the weights of a positive combination of normalized kernels using gradient descent. Bosch *et al* [3] learn the (level) weighting parameters of the spatial pyramid kernel using a validation set. The search for the optimal weighting is exhaustive, hence optimal, but may not scale as many kernels are added.

We see our contribution as follows: (i) We introduce SKM for object recognition. This technique – to our knowledge not previously used in vision – provides a tractable solution to the combinatorial kernel selection problem and allows the design of new kernels and features and ways to systematically assess their performance. (ii) We show that equivalent state-of-the-art results (marginally better or worse compared to existing methods based on insightful problem-dependent selection of kernel combinations) on large scale problems can be achieved with fewer, automatically selected kernels. We learn sparse kernel subsets for several datasets including Caltech 101 [6], the newly created Caltech 256 [8] and the pedestrian from INRIA [5] and report our experiences showing that SKM is a viable technology that improves and complements existing ones.

## 2. Support Kernel Machines

Although the fixed weighting of histogram intersections proposed in earlier work of [7, 13] make sense intuitively, we consider the more general problem of *learning* the weights instead. Ideally, we want to use several possibly inhomogeneous kernels and select the ones most effective for the task. In this section, we review techniques for learning multiple kernels – for details see [12, 1, 2].

Lanckriet *et al* [12] considered SVMs based on conic combinations of kernel matrices (linear combinations with non-negative coefficients) and showed that the optimization of the coefficients reduces to a convex optimization problem known as a quadratically-constrained quadratic program (QCQP). They provided an initial solution based on semi-definite programming, but that faced scalability problems for training sets larger than 2-3000 points. Bach *et al*.

[1] proposed a dual formulation of the QCQP as a second-order cone program, and showed how to exploit the technique of Moreau-Yosida regularization in order to yield a formulation to which SMO techniques can be applied. Because this method uses sparse regularizers to obtain a parsimonious conic combination of kernels, it is referred to as the *Support Kernel Machine (SKM)* [1]. More recent work by Bach *et al* [2] has shown that an entire regularization path can be efficiently computed for SKM using numerical continuation techniques. Unlike the regularization path of SVM which is piecewise linear, the one of SKM is piecewise smooth, each kink in the path corresponding to places where the pattern of sparsity in the linear combination of kernels changes. However, the main SKM optimization is performed using interior point (second-order) methods which makes it impractical for large problems. For our experiments, we follow [1].

**The Problem:** Assume we are given  $n$  data points  $(\mathbf{x}_i, y_i)$ , where  $\mathbf{x}_i$  is in the input space  $\mathcal{X} = \mathbb{R}^k$ , and  $y_i \in \{-1, 1\}$ . Consider the input space  $\mathcal{X}$  which will be mapped to  $m$  different feature spaces  $\mathcal{F}_1, \dots, \mathcal{F}_m$  using feature maps  $\Phi_1(\mathbf{x}), \dots, \Phi_m(\mathbf{x})$  and denote  $\Phi(\mathbf{x}) = (\Phi_1(\mathbf{x}), \dots, \Phi_m(\mathbf{x}))$  as the joint feature space. Consider also variables  $\mathbf{w}_i, i \in \{1 \dots m\}$  with joint vector  $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_m)$ . To encourage sparsity at the level of blocks, the cost function penalizes the block  $L_1$ -norm of  $\mathbf{w}$ . The multiple kernel learning problem is formulated as follows:

$$\min_{\mathbf{w} \in \mathcal{F}_1 \dots \times \mathcal{F}_m} \sum_{i=1}^n L(y_i \mathbf{w}^\top \Phi(\mathbf{x}_i)) + \lambda \left( \sum_{j=1}^m d_j \|\mathbf{w}_j\|_2 \right)^2 \quad (1)$$

where  $d_j$  are positive weights associated with each kernel,  $\lambda$  is a regularization constant, and  $L$  can be any loss function for classification, *e.g.* the hinge loss (see below). In the next section we give the basic primal problems for the simpler and more readable case of multiple linear classifiers, but the kernelization is straightforward [1].

### 2.1. Learning Multiple Linear Classifiers

To study the problem of learning a conic combination of linear classifiers, assume we are given a decomposition of  $\mathbb{R}^k$  as a product of  $m$  blocks:  $\mathbb{R}^k = \mathbb{R}^{k_1} \times \dots \times \mathbb{R}^{k_m}$ , so that each data point  $\mathbf{x}_i$  can be decomposed into  $m$  block components, *i.e.*  $\mathbf{x}_i = (\mathbf{x}_{1i}, \dots, \mathbf{x}_{mi})$ , where each  $\mathbf{x}_{ji}$  is a vector. The goal is to find a linear classifier of the form  $y = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$  where  $\mathbf{w}$  has the same block decomposition  $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_m) \in \mathbb{R}^{k_1 + \dots + k_m}$ . The primal for this problem is very similar to the one of a linear SVM, except for the cost used to penalize the coefficient, a block-weighted  $L_1$  norm, not an  $L_2$  norm, as usual:

$$\text{(P)} \quad \min \frac{1}{2} \left( \sum_{j=1}^m d_j \|\mathbf{w}_j\|_2 \right)^2 + C \sum_{i=1}^n \xi_i$$

$$\begin{aligned} \text{w.r.t. } & \mathbf{w} \in \mathbb{R}^{k_1} \times \dots \times \mathbb{R}^{k_m}, \boldsymbol{\xi} \in \mathbb{R}_+^n, b \in \mathbb{R} \\ \text{s.t. } & y_i \left( \sum_{j=1}^m \mathbf{w}_j^\top \mathbf{x}_{ji} + b \right) \geq 1 - \xi_i, \forall i \in \{1, \dots, n\} \end{aligned}$$

where a soft margin is used, with  $\xi_i$  slack variables.

The cost defined by **(P)** gives a convex yet non-differentiable dual problem. However, an  $L_2$  norm regularizer can be added to the cost in order to obtain a differentiable dual. This uses ‘bridge’ weightings  $a_j$  that are estimated during the optimization. Their role is to provide a dynamically adaptive cost that makes the problem locally smooth. The technique is known as Moreau-Yosida regularization [1]. The primal problem is:

$$\begin{aligned} \text{(RP)} \quad \min & \frac{1}{2} \left( \sum_{j=1}^m d_j \|\mathbf{w}_j\|_2 \right)^2 + \frac{1}{2} \sum_{j=1}^m a_j^2 \|\mathbf{w}_j\|_2^2 + C \sum_{i=1}^n \xi_i \\ \text{w.r.t. } & \mathbf{w} \in \mathbb{R}^{k_1} \times \dots \times \mathbb{R}^{k_m}, \boldsymbol{\xi} \in \mathbb{R}_+^n, b \in \mathbb{R} \\ \text{s.t. } & y_i \left( \sum_{j=1}^m \mathbf{w}_j^\top \mathbf{x}_{ji} + b \right) \geq 1 - \xi_i, \forall i \in \{1, \dots, n\} \end{aligned}$$

As typical with SVMs a first step to solve **(RP)** is to derive its Lagrangian. The saddle point, stationary conditions of the gradient, give a dual problem, which can be kernelized in the usual way by replacing each dot product with a kernel function [12, 1, 2].

### 3. Pyramid Match Kernels

In this section, we describe pyramid match kernels which inspired our study for automatically learning sparse combinations of kernels. Recently, Grauman & Darrell [7] and Lazebnik et al. [13] have successfully used histogram intersection kernels as an indirect means to approximate the number of point correspondences between two sets of features (images, say). Features are extracted in each image and separate histograms, with the same dimension are constructed. A histogram intersection function sums the minimum number of features in the corresponding bins of the two histograms, across all bins. The resulting pyramid match kernel is a weighted combination of histogram intersections which is itself a kernel.

Histogram intersection functions obtained in this way are positive-definite similarity functions [15], hence kernels. Both [7] and [13] define a pyramid of histogram functions as linear combinations of histogram intersections calculated coarse to fine. The weighting depends on the coarseness of the grid: histogram intersections at a coarser grid are overly penalized compared to intersections on finer grids. This is intuitively justified because matches found in larger cells / bins should be penalized more aggressively as they store

increasingly dissimilar features. The geometric weighting used in histogram intersections has a more formal justification, being known to approximate the optimal bipartite matching with good accuracy [7].

The difference between the kernels used by [7] and [13] comes in the way features are mapped to histograms. In [7], each dimension of the input feature vector is divided into  $2^l$ ,  $l = 0 \dots L$  equal sized bins. If the feature vector is in  $\mathbb{R}^d$ , the histogram dimension is  $D = 2^{ld}$ . A coarse grid corresponds to a lower value of  $l$  and a fine grid corresponds to a higher value of  $l$ . In [13], the grid is defined over the image, divided into  $2^l \times 2^l$  cells with features from spatially corresponding cells matched across any two images. We refer to the kernels obtained from the two methods using their acronyms: [7] as PMK and [13] as SPK.

### 4. Experiments

The experiments we show study the Caltech 101 and Caltech 256 datasets, as well as INRIA pedestrian, with a variety of pyramid match kernels, for both high and low regularization regimes. Intuitively, high regularization (this is the constant  $C$  in the primal problem) implies that we are enforcing a large margin separation between classes. This tends to activate more kernels especially when classes are not perfectly separated in the one-vs-all problems. Alternatively, low regularization values and easy, separable problems, will essentially lead to highly sparse solutions that use a very small number of kernels.

First we select  $N_{train}$  images from each class for training, e.g.  $N_{train} = 5, 10, 15, 20, 25, 30$ , and the remaining images in each class are used for testing. We learn a one-vs-all classifier for each class. A test image is classified by comparing probabilities assigned by each of the classifiers and assigning the class that has the highest probability. Classification performance on each class  $C$  is measured by determining the fraction of test examples from class  $C$  which are classified correctly. For Caltech 101 we train both using the sets of 4 kernels proposed by [7] and [13], and with an SKM based on a linear combination of all 8 kernels. We also train with an SKM based on these 8 plus 4 color feature based SPK type kernels which are explained in the following section. For Caltech 256 we train using only the set of 4 Spatial Pyramid Kernels [13]. For the INRIA pedestrian dataset we train using the sets of 4 kernels proposed by [7] and [13] but using a smaller negative training set than in [5]. Our primary goal is to understand under what circumstances learning kernels is useful and how does the learnt pattern of sparsity change with the difficulty of the problem.

**Kernels and Features:** We use our own implementation of the two types of kernels, PMK and SPK described in section 3, but with several minor differences that we briefly discuss.

We also describe the *color based kernel* that we used for Caltech 101.

We first describe our color based kernels used for Caltech 101. We bin the RGB values in a  $17 \times 17$  pixel patch into a  $4 \times 4 \times 4$  (64 dimensional) histogram and normalize by the number of pixels in the patch (289). This gives us a 64 dimensional feature. We extract these features on a grid with spacing of 8 pixels. Now the similarity between two images is measured using histogram intersection in the same way as SPK. We will refer to this kernel as *SPK-RGB*.

For PMK on the Caltech 101 dataset we first create an image pyramid of 10 scales, each a factor of  $2^{\frac{1}{4}}$  smaller than the previous image, using bicubic interpolation [14]. At each of the 10 scales we extract SIFT descriptors of  $41 \times 41$  pixel patches computed over a grid with spacing of 5 pixels. We project the 128 dimensional SIFT features to 10 dimensions using PCA to get the final feature representation. The dimensionality reduction is critical for good performance when using uniform bins for the feature histograms. We have four levels in the pyramid with 8, 4, 2 and 1 bins per dimension respectively. The first level is the finest (8 bins per dimension) and the fourth level is the coarsest (1 bin per dimension). For PMK on the INRIA pedestrian dataset and for SPK on all datasets, we use only the images at original scale. We extract SIFT descriptors of  $16 \times 16$  pixel patches computed on a grid with spacing of 8 pixels. We don't use the SIFT normalization procedure whenever the cumulative gradient magnitude of the patch is too weak [13]. For SPK we have four levels in the pyramid (compare to three in [13]) with grid sizes  $4 \times 4$ ,  $3 \times 3$ ,  $2 \times 2$  and  $1 \times 1$  respectively.

For each of the problems solved here, we consider the kernels from all the levels of the pyramid and learn the weights of the kernels [1]. We also vary the number of training images per class from 5 to 30 with a step of 5 examples. The pattern of sparsity we compute is shown using both binary and color coded diagrams (e.g. fig. 1). The number on the vertical indexes to the one-vs-all problem being trained (there are 101 for Caltech 101 and 256 for Caltech 256). The number on the horizontal indexes the kernel number from left to right, correspondingly fine to coarse levels in the pyramid (i.e.1 is finest and 4 is coarsest). Each rectangular bar corresponds to the kernel weights learnt for a particular number of positive training images per class, 5-30. The binary diagrams show which kernels are active (in black) and the color coded ones show their weights. We also compare the mean recognition rate obtained using learnt kernels with those using the geometric weighting suggested in [7] and [13] (which we call baseline results) (e.g. fig. 3). The geometric weighting for PMK [7] is (0.5, 0.25, 0.125, 0.125) from finest to coarsest level. The geometric weighting for SPK [13] is (0.5, 0, 0.25, 0.25) from finest to coarsest level (the second level corresponding to a  $3 \times 3$  spatial grid

does not exist in [13], hence the weight is zero).

#### 4.1. Caltech 101

**Pyramid Match Kernels (PMK) [7]:** In fig. 1 we show the sparsity pattern of PMK in the regime of high regularization, whereas the left plot in fig. 2 shows the regime at low regularization. For low regularization, only one kernel has positive weight for each one-vs-all classifier, hence only one plot. In fig. 3 we show classification accuracy results in both regimes compared to baseline results.

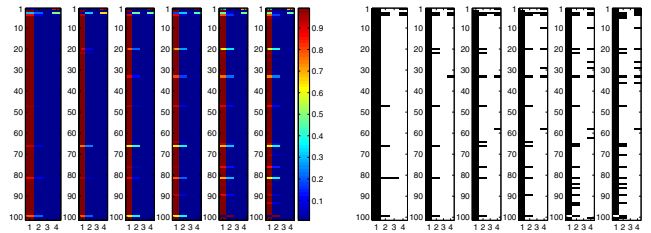


Figure 1. PMK (Caltech101): These plots are for high regularization. Left set of colored plots show kernel weights. Right set of black and white plots show sparsity pattern. Black means the corresponding kernel has a non-zero weight, white means it is turned off.  $N_{train} = 5, 10, 15, 20, 25, 30$  from left to right.

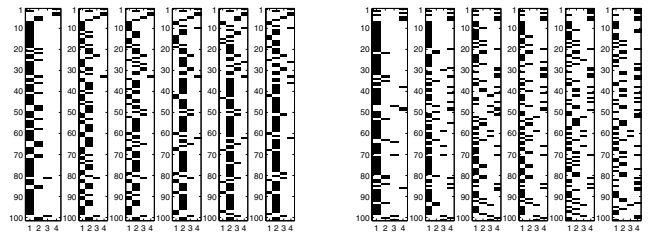


Figure 2. Caltech101: Sparsity of kernel weights for low regularization. (Left) PMK. (Right) SPK. Black means the corresponding kernel has a non-zero weight, white means it is turned off. Note that exactly one kernel is active for any given classifier. Also as the number of training images per class increases, the selected kernel tends to be the one for coarser level.  $N_{train} = 5, 10, 15, 20, 25, 30$  from left to right.

**Spatial Pyramid Kernels (SPK) [13]:** In fig. 4 we show the sparsity pattern of SPK in the regime of high regularization, whereas the right plot in fig. 2 shows the regime at low regularization. In fig. 5 we show classification accuracy results in both regimes compared to baseline results. We were not able to fully match the performance reported in [13]. One possible reason is the codebook construction, which is highly implementation dependent. Recent work by Jurie & Triggs [10] suggests that better results can be obtained by creating codebooks in which dense regions of the data are

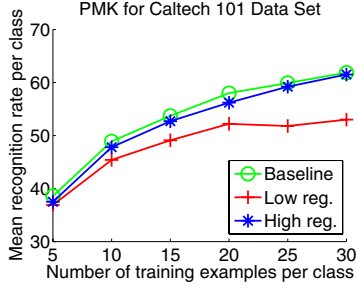


Figure 3. PMK (Caltech101): Mean recognition results on the Caltech 101 dataset for high and low values of the regularization parameter compared to baseline results. The quantitative values obtained were: *Baseline* 5 (38.6), 10 (48.9), 15 (53.7), 20 (58), 25 (59.9), 30 (61.8). *Low reg.*: 5 (36.9), 10 (45.4), 15 (49.1), 20 (52.2), 25 (51.8), 30 (53). *High regularization*: 5 (37.5), 10 (47.8), 15 (52.7), 20 (56.2), 25 (59.2), 30 (61.5).

effectively subsampled in order to avoid the excessive ‘attraction’ of centers. These narrow descriptor space regions, by the very virtue of their density, can be non-informative for classification.

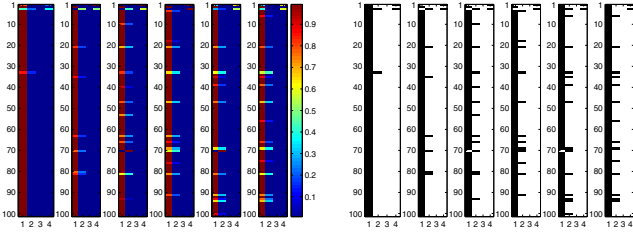


Figure 4. SPK (Caltech101): These plots are for high regularization. Left set of colored plots show kernel weights. Right set of black and white plots show sparsity pattern. Black means the corresponding kernel has a non-zero weight, white means it is turned off.  $N_{train} = 5, 10, 15, 20, 25, 30$  from left to right.

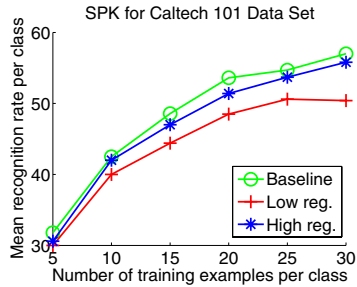


Figure 5. SPK: Recognition results on the Caltech 101 dataset for high and low values of the regularization parameter compared to baseline results. The quantitative values obtained were: *Baseline*: 5 (31.8), 10 (42.5), 15 (48.6), 20 (53.6), 25 (54.7), 30 (57). *Low regularization*: 5 (30), 10 (40), 15 (44.4), 20 (48.5), 25 (50.6), 30 (50.4). *High regularization*: 5 (30.6), 10 (42), 15 (47), 20 (51.4), 25 (53.7), 30 (55.8).

**PMK + SPK:** In fig. 6-7 we show weights learnt when the

entire 8 kernel set corresponding to PMK and SPK is considered. We show results for up to  $N_{train} = 5 - 20$ . Fig. 8 shows mean classification results for both high and low regularization regimes. Notice that the highest weighted kernel is usually the finest PMK kernel. This is consistent with the fact that in our implementation, PMK outperforms SPK. Overall, the sparse, learnt kernel subset outperforms each of SPK, PMK alone.

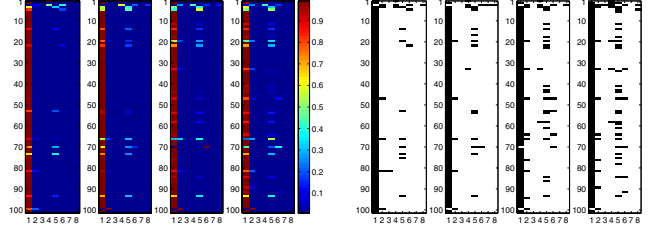


Figure 6. PMK+SPK (Caltech101): Training with 8 kernels, PMK first, SPK next in ordering. Left set of colored plots show kernel weights. Right set of black and white plots show sparsity pattern. Black means corresponding kernel has a non-zero weight, white means it is turned off.  $N_{train} = 5, 10, 15, 20$  from left to right.

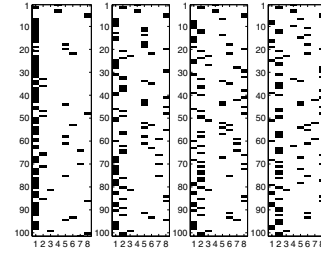


Figure 7. PMK+SPK (Caltech101): Sparsity of kernel weights for low regularization. Black means the corresponding kernel has a non-zero weight, white means it is turned off.  $N_{train} = 5, 10, 15, 20$  from left to right.

**PMK + SPK + SPK-RGB:** In fig. 9-10 we show weights learnt when the entire 12 kernel set corresponding to PMK, SPK and SPK-RGB is considered. We show results for up to  $N_{train} = 5 - 20$ . Fig. 11 shows mean classification results for both high and low regularization regimes. Overall, the sparse, learnt kernel subset outperforms each of SPK, PMK and SPK-RGB alone.

## 4.2. Caltech 256

For the Caltech 256 dataset we report results using the Spatial Pyramid Kernels (SPK) for classification (Pyramid Match Kernels are computationally more intensive). To map the extracted SIFT features to words we used the same codebook that was learnt from the Caltech 101 dataset (better results may be obtained using a Caltech 256 codebook).

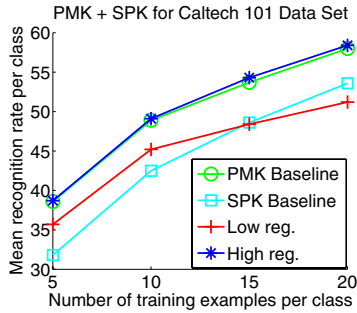


Figure 8. PMK+SPK (Caltech101): Recognition results on the Caltech 101 dataset for high and low values of the regularization parameter. *PMK Baseline*: 5 (38.6), 10 (48.9), 15 (53.7), 20 (58). *SPK Baseline*: 5 (31.8), 10 (42.5), 15 (48.6), 20 (53.6). *Low regularization*: 5 (35.7), 10 (45.2), 15 (48.4), 20 (51.2). *High regularization*: 5 (38.7), 10 (49.1), 15 (54.3), 20 (58.4). Clearly, automatic selection among all 8 kernels improves the results compared to any given quadruple in PMK or SPK.

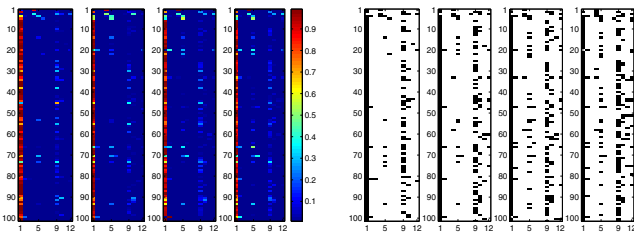


Figure 9. PMK + SPK + SPK-RGB (Caltech101): Training with 12 kernels, PMK, SPK and SPK-RGB in that order. Left set of colored plots show kernel weights. Right set of black and white plots show sparsity pattern. Black means the corresponding kernel has a non-zero weight, white means it is turned off.  $N_{train} = 5, 10, 15, 20$  from left to right.

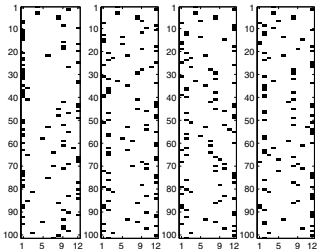


Figure 10. PMK + SPK + SPK-RGB (Caltech101): Sparsity of kernel weights for low regularization. Black means the corresponding kernel has a non-zero weight while white means it is turned off.  $N_{train} = 5, 10, 15, 20$  from left to right.

In fig. 12 we show the sparsity pattern of SPK in the regime of high regularization, whereas fig. 13 shows the regime at low regularization. We show results for varying number of training images per class (5, 10, 15, 20). Fig. 14 compares the overall classification accuracy for kernels combined using baseline weights [13] with those learnt using low and high regularization. To our knowledge, the only results re-

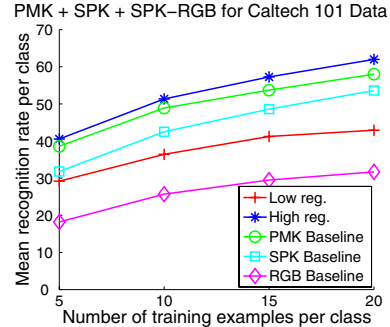


Figure 11. PMK + SPK + SPK-RGB (Caltech101): Recognition results on Caltech 101 dataset for high and low values of the regularization parameter. *PMK Baseline*: 5 (38.6), 10 (48.9), 15 (53.7), 20 (58). *SPK Baseline*: 5 (31.8), 10 (42.5), 15 (48.6), 20 (53.6). *SPK-RGB Baseline*: 5 (18.2), 10 (25.7), 15 (29.5), 20 (31.7). *Low regularization*: 5 (29.2), 10 (36.4), 15 (41.2), 20 (42.9). *High regularization*: 5 (40.5), 10 (51.3), 15 (57.3), 20 (62).

ported on Caltech 256 are by Griffin *et al* [8], who also used Spatial Pyramid Kernels [13]. With our sparse SPK kernel selection algorithm and the Caltech 101 codebook, [8] is marginally better. An interesting observation – by comparing the patterns of sparsity obtained for Caltech 101 and 256 – is that significantly more coarser kernels tend to be selected in the 256 classifiers in the low-regularization regime (see *e.g.* fig. 13). One possible explanation is the lack of centering in Caltech 256 [8] and finer levels no longer provide sufficient discriminative power when there is too much intraclass variability.

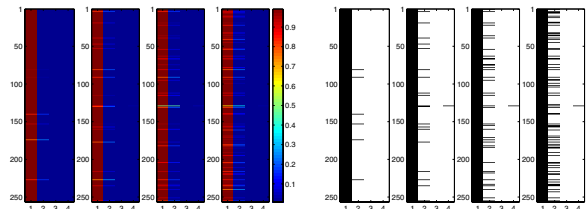


Figure 12. SPK (Caltech256): These plots are for high regularization. Left set of colored plots show kernel weights. Right set of black and white plots show sparsity pattern. Black means the corresponding kernel has a non-zero weight, white means it is turned off.  $N_{train} = 5, 10, 15, 20$  from left to right.

### 4.3. INRIA pedestrian

For INRIA pedestrian database [5] we use a set of 2172 positive and 2436 negative images for training. We test on 891 positive images and 765 negative images. We extract SIFT descriptors of  $16 \times 16$  pixel patches computed on a grid with spacing of 8 pixels and learn weights for four levels of SPK and PMK kernels.

For SPK, the weights learnt from finest level ( $4 \times 4$  grid) to coarsest (no grid) are (1, 0, 0, 0). We have four levels

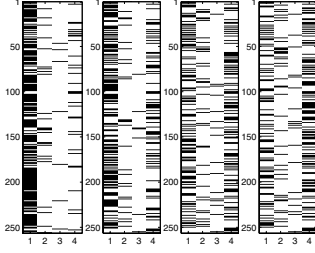


Figure 13. SPK (Caltech256): Sparsity of kernel weights for low regularization. Black means the corresponding kernel has a non-zero weight, white means it is turned off. Note that exactly one kernel is active for any given classifier.  $N_{train} = 5, 10, 15, 20$  from left to right.

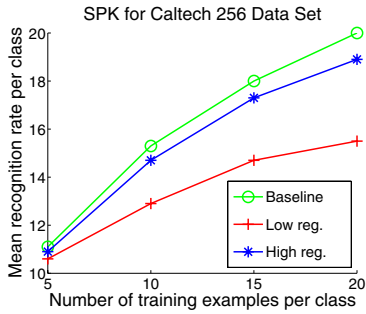


Figure 14. SPK (Caltech256): Recognition results on the Caltech 256 dataset for high and low values of the regularization parameter compared to baseline results. The quantitative values obtained were: *Baseline*: 5 (11), 10 (15.3), 15 (18), 20 (20). *Low regularization*: 5 (10.6), 10 (12.9), 15 (14.7), 20 (15.5). *High regularization*: 5 (10.9), 10 (14.7), 15 (17.3), 20 (18.9).

because we also consider a  $3 \times 3$  grid on the image. The classification accuracy on the positive test set is 99.3 % and 80 % on the negative test set. Using the weights of [13] (0.5, 0, 0.25, 0.25) the classification accuracy on the positive test set is 99.7 % and 69.5 % on the negative test set. To allow comparison with [5], we plot Detection Error Tradeoff (DET) curves on a log-log scale, *i.e.* miss rate (1-Recall or FalseNeg/(TruePos+FalseNeg)) versus False Positives Per Window (FPPW) in fig. 15. Lower values on DET curve are better. For PMK, the weights learnt from finest level ( $4 \times 4$  grid) to coarsest (no grid) are (0, 0, 1, 0). The classification accuracy on the positive test set is 89.2 % and 88.6 % on the negative test set. On the other hand, using the weights suggested in [7] (0.5, 0.25, 0.125, 0.125) the classification accuracy on the positive test set is 94.3 % and 91.1 % on the negative test set. We plot DET curves for fixed and learnt weights in fig. 16.

**Performance and running times:** The Support Kernel Machine algorithm we use [1] is based on SMO optimization, hence it is computationally efficient. If at any point during learning, only one kernel is selected, the method falls back on SVM-SMO. In our problems, the average running

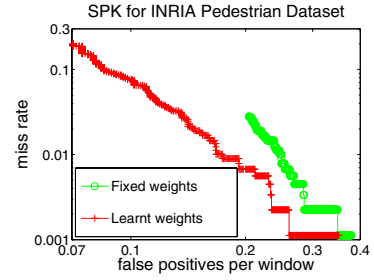


Figure 15. SPK: DET plots for fixed kernel weights (0.5, 0, 0.25, 0.25) and learnt weights (1, 0, 0, 0). The sparse, learnt kernel subset performs better.

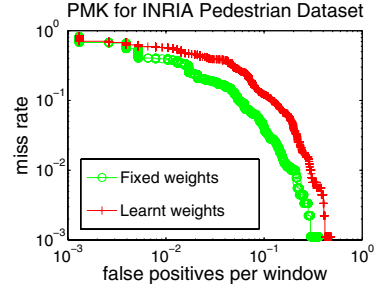


Figure 16. PMK: DET plots for fixed kernel weights (0.5, 0.25, 0.125, 0.125) and learnt weights (0, 0, 1, 0).

time for a 1-vs-all classification problem with, say, 15 positive training examples per class, is 38 seconds. The main computational burden remains the feature extraction and the computation of kernel matrices. Learning large sets of kernel combinations is memory intensive as the kernel matrices need to be stored, but caching techniques and predictive low-rank decomposition can be used to further improve performance.

#### 4.4. Discussion

One of the contributions of this paper is to show that parsimonious kernel combinations can be learnt in a tractable way using Support Kernel Machines. Consider, for example, the learnt patterns of sparsity for problems like the ones in fig. 9 corresponding to recognition results in fig. 11. Solutions of this form – a set of *different* kernels for each problem, but with good overall classification accuracy – are not easy to obtain using any of the algorithms currently used in object recognition. An SVM wrapper method faces a combinatorial problem and no simple kernel enumeration technique can solve it optimally. It is not surprising that learning kernels produces competitive state-of-the-art (marginally better or worse) classifiers, neither that a sparse combination may sometimes marginally hurt performance – this is a small price to pay for the benefit of compactness and selection. SKMs provide a scalable solution for combining large numbers of kernels with heterogeneous feature spaces, where a-priori weighting intuitions may no longer be available.

A second insight of our study is the somewhat limited performance of existing kernels on datasets like Caltech 101, 256 and INRIA pedestrian. We show negative experimental results implying that it is unlikely for combinations of existing kernels to produce significantly better results, at least within the span of our sparse representation and convex search problem. For low regularization values, most of our 1-vs-all classifiers achieved almost perfect separation on the training set. For this very reason their solutions are extremely sparse, often consisting of only one kernel. Once all hinge constraints are satisfied, the learner can only improve its cost by sparsifying the kernel combination, hence eliminating kernels. We often found that for many problems it is quite common that several kernels give very similar performance. In this case, the marginally best one will be selected (in most case this was the kernel computed on the finest grid). One way to turn on more kernels is by boosting the regularization constant – hence more heavily penalize errors at the margin. This can be a palliative for a marginally better solution, as long as perfect separation is still not completely achieved.

#### 4.5. Conclusions

We have argued that SVM classifiers based on linear combination of kernels can be a powerful way to boost the performance of the current recognition algorithms. However such classifiers introduced new difficulties, in particular the need to choose a weighting for the kernels, computational efficiency issues and increased propensity to overfit. The problem of learning a parsimonious set of kernels from a possibly large subset, falls beyond the methodology of current SVM. In this paper we advocate the transition from SVMs to Support Kernel Machines (SKM) in order to obtain models that estimate *both* the parameters of a *sparse* linear combination of kernels, *and* the parameters of a discriminative classifier in one convex problem. Our large scale study of representative datasets like Caltech 101, 256 or INRIA pedestrian show that state-of-the-art results can be achieved using sparse learnt kernel combinations (see *e.g.* fig. 11), but also underlines the limitations of current feature extraction and kernel design methods. In the long run, SKM appears to be a viable technique for designing new kernels and features, systematically assessing their performance, and selecting the best performing kernel subset for a given problem.

**Acknowledgements:** This work has been supported in part by the NSF and the EC, under awards IIS-0535140 and MCEXT-025481. We are grateful to Guillaume Obozinski for excellent SKM-SMO feedback and code.

#### References

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In

*ICML*, page 6, New York, NY, USA, 2004.

[2] F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In *NIPS*, 2004.

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proc. of the Int. Conf. on Image and Video Retrieval*, 2007.

[4] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, Washington, DC, USA, 2005.

[6] 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. In *CVPR, Workshop on Generative-Model Based Vision*, 2004.

[7] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465, Washington, DC, USA, 2005.

[8] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report UCB/CSD-04-1366, California Institute of Technology, 2007.

[9] A. Holub and P. Perona. A discriminative framework for modelling object classes. In *CVPR*, pages 664–671, Washington, DC, USA, 2005.

[10] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.

[11] F. D. la Torre Frade and O. Vinyals. Learning kernel expansions for image classification. In *CVPR*, June 2007.

[12] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

[13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, Washington, DC, USA, 2006.

[14] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, pages 11–18, Washington, DC, USA, 2006.

[15] F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *IEEE Trans. on Image Processing*, 14(2), Feb 2005.

[16] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *NIPS 19*. MIT Press, Cambridge, MA, 2007.

[17] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, pages 994–1000, Washington, DC, USA, 2005.

[18] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, pages 1331–1338, Washington, DC, USA, 2005.

[19] H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, pages 2126–2136, Washington, DC, USA, 2006.