Multiple Kernel Learning for Visual Object Recognition: A Review

Serhat S. Bucak, *Student Member, IEEE,* Rong Jin, *Member, IEEE,*
and Anil K. Jain, *Fellow, IEEE*

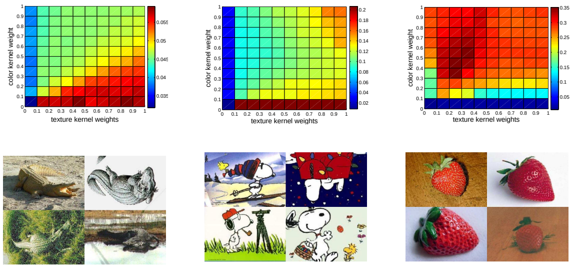
Abstract—Multiple kernel learning (MKL) is a principled approach for selecting and combining kernels for a given recognition task. A number of studies have shown that MKL is a useful tool for object recognition, where each image is represented by multiple sets of features and MKL is applied to combine different feature sets. We review the state-of-the-art for MKL, including different formulations and algorithms for solving the related optimization problems, with the focus on their applications to object recognition. One dilemma faced by practitioners interested in using MKL for object recognition is that different studies often provide conflicting results about the effectiveness and efficiency of MKL. To resolve this, we conduct extensive experiments on standard datasets to evaluate various approaches to MKL for object recognition. We argue that the seemingly contradictory conclusions offered by studies are due to different experimental setups. The conclusions of our study are, (i) given a sufficient number of training examples and feature/kernel types, MKL is more effective for object recognition than simple kernel combination (e.g., choosing the best performing kernel or average of kernels), and (ii) among the various approaches proposed for MKL, the sequential minimal optimization, semi-infinite programming, and level method based ones are computationally most efficient.

Index Terms—Multiple kernel learning, support vector machine, visual object recognition, convex optimization.

1 Introduction

Kernel methods [1] have become popular in computer vision, particularly for visual object recognition. The key idea of kernel methods is to introduce nonlinearity into the decision function by mapping the original features to a higher dimensional space. Many studies [2], [3], [4] have shown that nonlinear kernels, such as radial basis functions (RBF) or chi-squared kernels, yield significantly higher accuracy for object recognition than a linear classification model.

One difficulty in developing kernel classifiers is to design an appropriate kernel function for a given task. We often have multiple kernel candidates for visual object recognition. These kernels arise either because multiple feature representations are derived for images, or because different kernel functions (e.g., polynomial, RBF, and chi-squared) are used to measure the visual similarity between two images for a given feature representation. One of the key challenges in visual object recognition is to find the optimal combination of these kernels for a given object class. This is the central question addressed by Multiple Kernel Learning (MKL).



crocodile snoopy strawberry

Fig. 1: The first row shows the surface graphs that demonstrate the influence of different kernel combination weights on the mean average precision score for three different classes. Four examples from each class are given in the second row.

We motivate the use of MKL on a simple object recognition problem. We create two kernels: one based on color histogram and one based on texture distribution in the image. We choose three object

- *The authors are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824.*
- *A.K. Jain is also affiliated with the Department of Brain and Cognitive Engineering, Korea University, Seoul. All correspondence regarding this manuscript should be addressed to A.K. Jain.*
E-mail: {bucakser,rongjin,jain}@cse.msu.edu

classes (crocodile, snoopy, strawberry) from Caltech 101 dataset [5], each with 15 instances, and train one-vs-all SVM for each of the three classes by using different combinations of these two kernels. To combine the kernels, we vary the combination coefficients in the set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. In Figure 1 we generate a heat map to represent classification performance of different linear combinations of the two kernels. We observe that the optimal combination varies from one class to other. For example, while texture based kernel is assigned a higher coefficient for crocodile classification task, color feature should be used with a higher weight for the strawberry class. This simple example illustrates the significance of identifying appropriate combination of multiple kernels for recognizing a specific class of visual objects. It also motivates the need for developing automatic approaches for finding the optimal combination of kernels from training examples as there is no universal solution for kernel combination that works well for all visual object classes.

MKL has been successfully applied to a number of tasks in computer vision. For instance, the winning group in Pascal VOC 2010 object categorization challenge [3] used MKL to combine multiple sets of visual features. The best performance reported on the Caltech 101 dataset was achieved by learning the optimal combination of multiple kernels [6]. Recent studies have also shown promising performance of MKL for object detection [7].

There is a considerable amount of literature on MKL. While many of the studies address the efficiency and effectiveness of MKL methods [8], [9], [10], [11], [12], [13], a large number of studies focus on the application of MKL to different domains [14], [7], [15]. Most published MKL studies are limited in their scope in the sense that only a small subset of MKL algorithms are compared under a limited number of experimental conditions, making it difficult to generalize their conclusions.

A lack of comprehensive studies has resulted in different, sometimes conflicting, statements regarding the effectiveness of various MKL methods on real-world problems. For instance, some of the studies [6], [22], [13], [8] reported that MKL outperforms the average kernel baseline (assigning equal weights to all the kernels) while other studies made the opposite conclusion [30], [31], [24] (see Table 1). Moreover, as Table 2 shows, there are also some confusing results and statements about the efficiency of different MKL methods. The problem of having discordant conclusions drawn on the empirical performance of different MKL methods motivated us to write this review article. Besides summarizing the latest developments in MKL and its application to visual object recognition, an important contribution of this paper is to resolve the conflicting statements by conducting a comprehensive evaluation of state-of-the-art MKL algorithms under various experimental conditions. In this survey, we focus on visual object recognition because among all the applications of MKL in computer vision, it is in object recognition domain that MKL has been reported to achieve the greatest success.

The main contributions of this survey are:

- A review of a wide range of MKL formulations that use different regularization mechanisms, and the related optimization algorithms.
- A comprehensive study that evaluates and compares a representative set of MKL algorithms for visual object recognition under different experimental settings.
- An exposition of the conflicting statements regarding the performance of different MKL methods, particularly for visual object recognition. We attempt to understand these statements and determine to what degree and under what conditions these statements are correct.

There does exist a survey of MKL published in 2011 [32]. However, the focus of our work is very different from [32]. First, while [32] provides a general review of MKL, we choose to focus on the application of MKL to visual object recognition, a task where MKL has shown great success. Second, while the empirical studies in [32] are based on small UCI datasets and focus on comparing different MKL formulations, we conduct experiments on significantly larger object recognition datasets.

2 Overview

In this section we give an overview of multiple kernel learning. We also briefly describe visual object recognition task as an application of MKL.

2.1 Overview of Multiple Kernel Learning (MKL)

MKL was first proposed in [33], where it was cast into a Semi-Definite Programming (SDP) problem. Most studies on MKL are centered around two issues, (i) how to improve the classification accuracy

TABLE 1: Comparison of MKL baselines and simple baselines (“Single” for single best performing kernel and “AVG” for the average of all the base kernels) in terms of classification accuracy. The last three columns give the references in which either “method1” or “method2” performs better, or both methods give comparable results, respectively.

method1	**method2**	**dataset**	**# samples**	**# kernels**	**method1 better**	**method2 better**	**similar**
MKL	Single	UCI	[1-6,000]	[1-10]	[16], [17]		[18], [19]
MKL	Single	UCI	[1-2,000]	[10-200]	[16], [17]	[20]	[20]
L_1-MKL	AVG	Caltech 101	[510-3,060]	[10-1,000]	[21], [22]	[23],[24]	[25], [13]
L_1-MKL	AVG	VOC07	5011	[10-22]	[22],	[13]	[26]
L_1-MKL	AVG	Oxford Flowers	680	[5-65]			[25], [23]
L_p-MKL	AVG	VOC07	5011	10	[26]		
L_p-MKL	AVG	Caltech 101	[1,530-3,060]	[24-1,000]	[13]		[24]
L_p-MKL	AVG	Oxford Flowers	680	[5,65]			[13]
L_1-MKL	L_p-MKL	UCI	[1-2,000]	[1-50]	[27]	[28], [8]	[29]
L_1-MKL	L_p-MKL	VOC07	5011	[10-22]		[26],[13]	
L_1-MKL	L_p-MKL	Caltech 101	[510-3,060]	[10-1,000]		[24]	[13]

of MKL by exploring different formulations, and (ii) how to improve the learning efficiency of MKL by exploiting different optimization techniques (see Fig. 2).

In order to learn an appropriate kernel combination, various regularizers have been introduced for MKL, including L_1 norm [34], L_p norm ($p > 1$) [10], entropy based [30], and mixed norms [35]. Among them, L_1 norm is probably the most popular choice because it results in sparse solutions and could potentially eliminate irrelevant and noisy kernels. In addition, theoretical studies [36], [37] have shown that L_1 norm will result in a small generalization error even when the number of kernels is very large.

A number of empirical studies have compared the effect of different regularizers used for MKL [8], [38], [13]. Unfortunately, different studies arrive at contradictory conclusions. For instance, while many studies claim that L_1 regularization yields good performance for object recognition [11], [24], others show that L_1 regularization results in information loss by imposing sparseness over MKL solutions, thus leading to suboptimal performance [8], [30], [38], [13], [39] (see Table 1).

In addition to a linear combination of base kernels, several algorithms have been proposed to find a nonlinear combination of base kernels [12], [28], [44], [23], [45]. Some of these algorithms try to find a polynomial combination of the base kernels [12], [28], while others aim to learn an instance-dependent linear combination of kernels [46], [47], [6]. The main shortcoming of these approaches is that they have to deal with non-convex optimization problems, leading to poor computational efficiency and suboptimal performance. Given these shortcomings, we will not review them in detail.

Despite significant efforts in improving the effectiveness of MKL, one of the critical questions remaining is whether MKL is more effective than a simple baseline, e.g., taking the average of the base kernels. While many studies show that MKL algorithms bring significant improvement over the average kernel approach [8], [39], [48] , opposite conclusions have been drawn by some other studies [30], [13], [31], [24]. Our empirical studies show that these conflicting statements are largely due to the variations in the experimental conditions, or in other words, the consequence of a lack of comprehensive studies on MKL.

The second line of research in MKL is to improve the learning efficiency. Many efficient MKL algorithms [34], [49], [43], [8], [30], [9], [44] have been proposed, mostly for L_1 regularized MKL, based on the first order optimization methods. We again observe conflicting statements in the MKL literature when comparing different optimization algorithms. For instance, while some studies [8], [42], [41] report that the subgradient descent (SD) algorithms [43] are more efficient in training MKL than the semi-infinite linear programming (SILP) based algorithm [50], an opposing statement was given in [11]. It is important to note that besides the training time, the sparseness of the solution also plays an important role in computational efficiency: both the number of active kernels and the number of support vectors affect the number of kernel evaluations and, consequentially, computational times for both training and testing. Unfortunately, most studies focus on only one aspect of computational efficiency: some only report the total training time [30], [11] while others focus on the number of

TABLE 2: Comparison of computational efficiency of MKL methods. The last three columns give the references, where “method1” is better, “method2” is better, or both give similar results.

eval. criterion	method1	method2	datasets	# samples	# kernels	method1 better	method2 better	similar
training time	L_1-MKL	L_p-MKL	Mediamill	30,993	3			[40]
training time	MKL-L1	L_p-MKL	UCI	[1-2,000]	[90-800]	[30]		
training time	MKL-SD	MKL-SIP	UCI datasets	[1-2,000]	[50-200]	[41],[42],[43],[8]		
training time	MKL-SD	MKL-SIP	Oxford Flowers	680	[5-65]		[25]	
# active kernels	MKL-SD	MKL-SIP	UCI datasets	[1-2,000]	[50-200]		[41],[42],[43]	
training time	MKL-SD	MKL-MD	Oxford Flowers	680	[5-65]		[23]	
training time	MKL-SD	MKL-MD	Caltech 101	3,060	9	[22]		
training time	MKL-SD	MKL-MD	VOC07	5,011	22		[22]	
training time	MKL-SD	MKL-Level	UCI datasets	[1-2,000]	[50-200]	[42]		
# active kernels	MKL-SD	MKL-Level	UCI datasets	[1-2,000]	[50-200]		[42]	
training time	MKL-SIP	MKL-Level	UCI datasets	[1-2,000]	[50-200]		[42]	
# active kernels	MKL-SIP	MKL-Level	UCI datasets	[1-2,000]	[50-200]	[42]		

support vectors (support set size) [47], [8]. Another limitation of the previous studies is that they are mostly constrained to small datasets (around 1,000 samples) and limited number of base kernels (10 to 50), making it difficult to draw meaningful conclusions on the computational efficiency. One goal of this article is to provide a comprehensive examination of computational efficiency for MKL from the viewpoints of training time and solution sparseness.

2.2 Relationship to the Other Approaches

Multiple kernel learning is closely related to feature selection [51], where the goal is to identify a subset of features that are optimal for a given prediction task. This is evidenced by the equivalence between MKL and group lasso [52], a feature selection method where features are organized into groups, and the selection is conducted at the group level instead of at the level of individual features.

MKL is also related to metric learning [53], where the goal is to find a distance metric, or more generally a distance function, consistent with the class assignment. MKL generalizes metric learning by searching for a combination of kernel functions that gives a larger similarity to any instance pair from the same class than instance pairs from different classes.

Finally, it is important to note that multiple kernel learning is a special case of kernel learning. In addition to MKL, another popular approach for learning a linear combination of multiple kernels is kernel alignment [54], which finds the optimal combination of kernels by maximizing the alignment between the combined kernel and the class assignments of training examples. More generally, kernel learning methods can be classified into two groups: parametric and non-parametric kernel learning. In parametric kernel learning, a parametric form is assumed for the combined kernel function [55], [56]. In contrast, nonparametric kernel learning does not make any parametric assumption about the target kernel function [57], [58], [54]. Multiple kernel learning belongs to the category of parametric kernel learning. Despite its generality, the high computational cost of non-parametric kernel learning limits its applications to real-world problems. Aside from supervised kernel learning, both semi-supervised and unsupervised kernel learning have also been investigated [54], [56], [59]. We do not review them in detail here because of their limited success in practice and because of their high computational cost.

2.3 Visual Object Recognition

The goal of visual object recognition is to determine if a given image contains one or more objects belonging to the class of interest (without necessarily locating the position of the object in the image). It is a challenging task because of the large variations in the visual appearance of objects from the same class [60], [61] caused by viewpoint variation and occlusion, articulation, background clutter, illumination, and pose change.

Among various approaches developed for visual object recognition, the bag-of-words (BoW) model is probably the most popular due to its simplicity and success in practice. There are numerous options for each step of the BoW model. In this section, we will briefly state these steps and discuss the options for each step that we use in our experiments to construct the base kernels.

The first step of the BoW model is to detect keypoints or keyregions from images. Many algorithms have been developed for keypoint/region detection [62], [63], [64], each having its own strength and weakness. For instance, although dense sampling is shown to be superior to other techniques for object recognition, it usually yields a large number of keypoints and might lead to high computational costs. To have a richer variety of representations, in our experiments we used Harris-Laplacian [63] and canny-edge detector based keypoint detection methods in addition to dense sampling. The second step is to generate local descriptors for the detected keypoints/regions. There is a rich literature on local descriptors, among which scale invariant feature transform (SIFT) [64] is, without doubt, the most popular. Other techniques we use in our experiments to improve the recognition performance are local binary patterns (LBP) [65] and histogram of oriented gradients (HOG) [66].

Given the computed descriptors, the third step of the BoW model is to construct the visual vocabulary. Both the dictionary size and the technique used to create the dictionary can have significant impact on the final recognition accuracy. In our experiments, we use k-means clustering technique to generate the dictionary. Given the dictionary, the next step is to map each keypoint to a visual word in the dictionary, a step that is often referred to as the encoding module. Recent studies express a vast amount of interest in the encoding step, resulting in many alternatives to vector quantization (e.g., Fisher kernel representation [67]).

The last step of the BoW model is the pooling step that pools encoded local descriptors into a global histogram representation. Various pooling strategies have been proposed for the BoW model such as mean and max-pooling, two techniques that we employ in our experiments. Studies [68] have shown that it is important to take into account the spatial layout of keypoints in the pooling step. One common approach is to divide an image into multiple regions and construct a histogram for each region separately. A well known example of this approach is spatial pyramid pooling [68] that divides an image into 1×1, 2×2, and 4×4 grids.

Besides the BoW model, many alternative image features have been proposed for object recognition, including GIST [69], color histograms, V1S+ [70] and geometric blur [71]. As demonstrated in our experiments, these techniques can be successfully combined with the BoW model for a better image representation. Given these additional alternatives and the large number of ways for constructing the BoW model, one critical issue in developing statistical models for visual object recognition is how to effectively combine different image representations for accurate object recognition. MKL presents a principled framework for combining multiple image representations: it creates a base kernel for each representation and finds the optimal kernel combination via a linear combination of kernels.

3 Multiple Kernel Learning (MKL): Formulations

In this section, we first review the theory of multiple kernel learning for binary classification. We then discuss MKL for multi-class multi-label learning.

3.1 MKL for Binary Classification

Let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a collection of n training instances, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a compact domain. Let $\mathbf{y} = (y_1, \ldots, y_n)^\top \in \{-1, +1\}^n$ be the vector of class assignments for the instances in $\mathcal{D}$. We denote by $\{\kappa_j(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R},\ j = 1, \ldots, s\}$ the set of s base kernels to be combined. For each kernel function $\kappa_j(\cdot, \cdot)$, we construct a kernel matrix $\mathbf{K}_j = [\kappa_j(\mathbf{x}_i, \mathbf{x}_{i'})]_{n \times n}$ by applying $\kappa_j(\cdot, \cdot)$ to the training instances in $\mathcal{D}$. We denote by $\boldsymbol{\beta} = (\beta^1, \ldots, \beta^s)^\top \in \mathbb{R}^s_+$ the set of coefficients used to combine the base kernels, and denote by $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \kappa_j(\mathbf{x}, \mathbf{x}')$ and $\mathbf{K}(\boldsymbol{\beta}) = \sum_{j=1}^s \beta_j \mathbf{K}_j$ the combined kernel function and kernel matrix, respectively. We further denote by $\mathcal{H}_\beta$ the Reproducing Kernel Hilbert Space (RKHS) endowed with the combined kernel $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta})$. In order to learn the optimal combination of kernels, we first define the regularized classification error $\mathcal{L}(\boldsymbol{\beta})$ for a combined kernel $\kappa(\cdot, \cdot; \boldsymbol{\beta})$, i.e.

$$\mathcal{L}(\boldsymbol{\beta}) = \min_{f \in \mathcal{H}_\beta} \frac{1}{2} ||f||^2_{\mathcal{H}_\beta} + C \sum_{i=1}^n \ell(y_i f(\mathbf{x}_i)), \quad (1)$$

where $\ell(z) = \max(0, 1 - z)$ is the hinge loss and $C > 0$ is a regularization parameter. Given the regularized classification error, the optimal combination vector $\boldsymbol{\beta}$ is found by minimizing $\mathcal{L}(\boldsymbol{\beta})$, i.e.

$$\min_{\beta \in \Delta, f \in \mathcal{H}_\beta} \frac{1}{2} ||f||^2_{\mathcal{H}_\beta} + C \sum_{i=1}^n \ell(y_i f(\mathbf{x}_i)) \quad (2)$$

where Δ is a convex domain for combination weights $\boldsymbol{\beta}$ that will be discussed later. As in [33], the problem in (2) can be written into its dual form, leading to the following convex-concave optimization problem

$$\min_{\beta\in\Delta}\max_{\alpha\in\mathcal{Q}} \widehat{\mathcal{L}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y}), \tag{3}$$

where $\circ$ denotes the Hadamard (element-wise) product, $\mathbf{1}$ is a vector of all ones, and $\mathcal{Q} = \{\boldsymbol{\alpha} \in [0, C]^n\}$ is the domain for dual variables $\boldsymbol{\alpha}$.

The choice of domain Δ for kernel coefficients can have a significant impact on both classification accuracy and efficiency of MKL. One common practice is to restrict $\boldsymbol{\beta}$ to a probability distribution, leading to the following definition of domain Δ [33], [34],

$$\Delta_1 = \left\{ \boldsymbol{\beta} \in \mathbb{R}_+^s : \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^s |\beta^j| \leq 1 \right\}. \tag{4}$$

Since Δ_1 bounds $\|\boldsymbol{\beta}\|_1$, we also refer to MKL using Δ_1 as the L_1 regularized MKL, or L_1-MKL. The key advantage of using Δ_1 is that it will result in a sparse solution for $\boldsymbol{\beta}$, leading to the elimination of irrelevant kernels and consequentially an improvement in computational efficiency as well as robustness in classification.

The robustness of L_1-MKL is verified by the analysis in [36], which states that the additional generalization error caused by combining multiple kernels is $O(\sqrt{\log s/n})$ when using Δ_1 as the domain for $\boldsymbol{\beta}$, implying that L_1-MKL is robust to the number of kernels as long as the number of training examples is not too small. The advantage of L_1-MKL is further supported by the equivalence between L_1-MKL and feature selection using group Lasso [52]. Since group Lasso is proved to be effective in identifying the groups of irrelevant features, L_1-MKL is expected to be resilient to weak kernels.

Despite the advantages of L_1-MKL, it was reported in [40] that sparse solutions generated by L_1-MKL might result in information loss and consequentially suboptimal performance. As a result, L_p regularized MKL (L_p-MKL), with $p > 1$, was proposed in [11], [10] in order to obtain a smooth kernel combination, with the following definition for domain Δ

$$\Delta_p = \left\{ \boldsymbol{\beta} \in \mathbb{R}_+^s : ||\boldsymbol{\beta}||_p \leq 1 \right\}. \tag{5}$$

Among various choices of L_p-MKL ($p > 1$), L_2-MKL is probably the most popular one [40], [31], [10]. Other smooth regularizers proposed for MKL include negative entropy (i.e., $\sum_{k=1}^s \beta_k \log \beta_k$) [30] and Bregman divergence [9]. In addition, hybrid approaches have been proposed to combine different regularizers for MKL [15], [31], [72].

Although many studies compared L_1 regularization to smooth regularizers for MKL, the results are inconsistent. While some studies claimed that L_1 regularization yields better performance for object recognition [11], [24], others show that L_1 regularization may result in suboptimal performance due to the sparseness of the solutions [8], [30], [38], [13], [39]. In addition, some studies reported that training an L_1-MKL is significantly more efficient than training a L_2-MKL [30], while others claimed that the training times for both MKLs are comparable [40].

A resolution to these contradictions, as revealed by our empirical study, depends on the number of training examples and the number of kernels. In terms of classification accuracy, smooth regularizers are more effective for MKL when the number of training examples is small. Given a sufficiently large number of training examples, particularly when the number of base kernels is large, L_1 regularization is likely to outperform the smooth regularizers.

In terms of computation time, we found that L_p-MKL methods are generally more efficient than L_1-MKL. This is because the objective function of L_p-MKL is smooth while the objective function of L_1-MKL is not [1]. As a result, L_p-MKL enjoys a significantly faster convergence rate ($O(1/T^2)$) than L_1-MKL ($O(1/T)$) according to [73], where T is the number of iterations. However, when the number of kernels is sufficiently large and kernel combination becomes the dominant computational cost at each iteration, L_1-MKL can be as efficient as L_p-MKL because L_1-MKL produces sparse solutions.

One critical question that remains to be answered is whether a linear MKL is more effective than simple approaches for kernel combination, e.g. using the best single kernel (selected by cross validation) or the average kernel method. Most studies show that L_1-MKL outperforms the best performing kernel, although there are scenarios where kernel combination might not perform as well as the single best performing kernel [40]. Regarding the comparison of MKL to the average kernel baseline, the answer is far from conclusive (see Table 1). While some studies show that L_1-MKL brings significant improvement over the average kernel approach [8],

[1] A function is smooth if its gradient is Lipschitz continuous

[39], [48], other studies claim the opposite [30], [13], [31], [24]. As revealed by the empirical study presented in Section 5, the answer to this question depends on the experimental setup. When the number of training examples is not sufficient to identify the strong kernels, MKL may not perform better than the average kernel approach. But, with a large number of base kernels and a sufficiently large number of training examples, MKL is very likely to outperform, or at least yield similar performance as, the average kernel approach.

Besides the linear kernel combination, several algorithms have been proposed to learn the nonlinear combination of multiple kernels from training data [45], [28], [12], [17], [47], [6], [46]. We skip these methods because of their limited success and high computational cost.

3.2 Multi-class and Multi-label MKL

A straightforward approach for multi-label MKL (ML-MKL) is to decompose a multi-label learning problem into a number of binary classification tasks using either one-versus-all (OvA) or one-versus-one (OvO) approaches. Tang et al. [74] evaluated three different strategies for multi-label MKL based on the OvA and concluded that learning one common kernel combination shared by all classes not only is efficient but also yields classification performance that is comparable to choosing different kernel combinations for different classes. This result is further confirmed in [22]. The main advantage of using the shared kernel combination is its computational efficiency. Although the assumption that all classifiers share the same kernel combination may not hold in general, it seems to work well for object recognition according to the empirical study in [22].

4 Multiple Kernel Learning: Optimization Techniques

A large number of algorithms have been proposed to solve the optimization problems posed in (2) and (3). We can broadly classify them into two categories. The first group of approaches directly solve the primal problem in (2) or the dual problem in (3). We refer to them as the *direct approaches*. The second group of approaches, solves the convex-concave optimization problem in (3) by alternating between two steps, i.e., the step for updating the kernel combination weights and the step for solving the SVM classifier for the given combination weights. We refer to them as the *wrapper approaches*. Fig. 2 summarizes different optimization methods developed for MKL. We note that due to the scalability issue, almost all the MKL algorithms are based on the first order method (i.e. iteratively updates the solution based on the gradient of the objective function or the most violated constraint). We refer the readers to [42], [38], [75] for more discussion about the equivalence or similarities among different MKL algorithms.

4.1 Direct Approaches for MKL

Lanckriet et al. [33] showed that the problem in (2) can be cast into Semi-Definite Programming (SDP) problem, i.e.,

$$\begin{aligned} \min_{\mathbf{z}\in\mathbb{R}^n,\boldsymbol{\beta}\in\Delta,t\geq 0} \quad & t/2 + C\sum_{i=1}^{n}\max(0, 1-y_i z_i) \\ \text{s. t.} \quad & \begin{pmatrix} \mathbf{K}(\boldsymbol{\beta}) & \mathbf{z} \\ \mathbf{z}^\top & t \end{pmatrix} \succeq 0 \end{aligned} \tag{6}$$

Although general-purpose optimization tools such as SeDuMi [76] and Mosek [77] can be used to directly solve the optimization problem in (6), they are computationally expensive and are unable to handle more than a few hundred training examples.

Besides directly solving the primal problem, several algorithms have been developed to directly solve the dual problem in (3). Bach et al. [16] propose to solve the dual problem using the technique of sequential minimal optimization (SMO) [78]. In [30], the authors applied the Nesterov's method to solve the optimization problem in (3). Although both approaches are significantly more efficient than the direct approaches that solve the primal problem of MKL, they are generally less efficient than the wrapper approaches [34].

4.1.1 A Sequential Minimum Optimization (SMO) based Approach for MKL

This approach is designed for L_p-MKL. Instead of constraining $\|\beta\|_p \leq 1$, Vishwanathan et al. proposed to solve a regularized version of MKL in [9], and converted it into the following optimization problem,

$$\max_{\boldsymbol{\alpha}\in Q} \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{8\lambda}\left(\sum_{k=1}^{s}\left[(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}_k(\boldsymbol{\alpha}\circ\mathbf{y})\right]^q\right)^{\frac{2}{q}}. \tag{7}$$

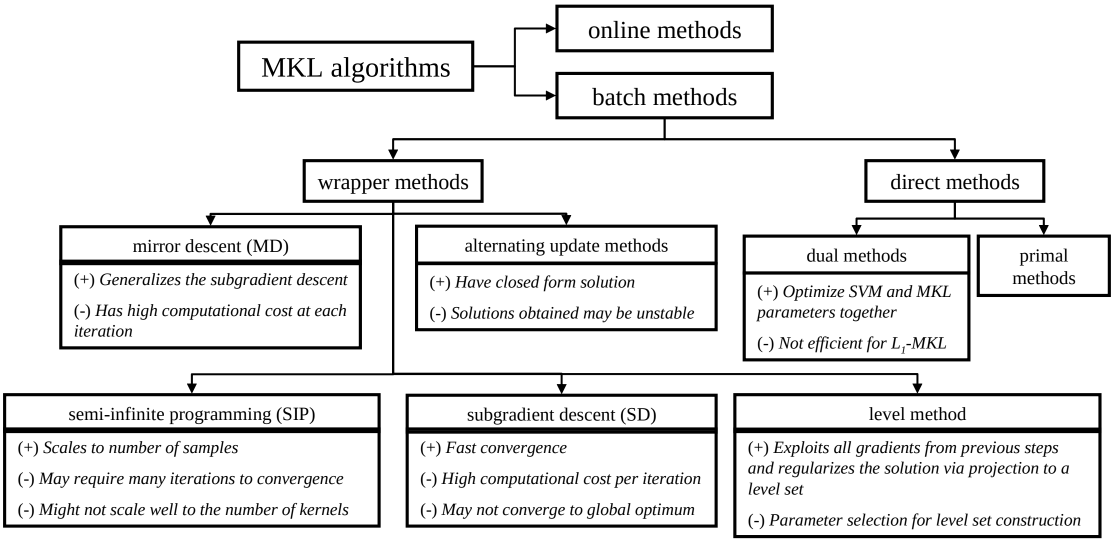

MKL algorithms
online methods
batch methods
wrapper methods
direct methods
mirror descent (MD)
(+) Generalizes the subgradient descent
(-) Has high computational cost at each iteration
alternating update methods
(+) Have closed form solution
(-) Solutions obtained may be unstable
dual methods
(+) Optimize SVM and MKL parameters together
(-) Not efficient for L_1-MKL
primal methods
semi-infinite programming (SIP)
(+) Scales to number of samples
(-) May require many iterations to convergence
(-) Might not scale well to the number of kernels
subgradient descent (SD)
(+) Fast convergence
(-) High computational cost per iteration
(-) May not converge to global optimum
level method
(+) Exploits all gradients from previous steps and regularizes the solution via projection to a level set
(-) Parameter selection for level set construction

Fig. 2: A summary of representative MKL optimization schemes

It can be shown that given $\boldsymbol{\alpha}$, the optimal solution for $\boldsymbol{\beta}$ is given by

$$\beta_j = \frac{\gamma_j}{2\lambda}\left(\sum_{k=1}^{s}\left((\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}_k(\boldsymbol{\alpha}\circ\mathbf{y})\right)^q\right)^{\frac{1}{q}-\frac{1}{p}} \tag{8}$$

where $\gamma_j = \left((\boldsymbol{\alpha}\circ\mathbf{y})^\top K_j(\boldsymbol{\alpha}\circ\mathbf{y})\right)^{\frac{q}{p}}$ and $q^{-1}+p^{-1}=1$. Since the objective given in (7) is differentiable, a Sequential Minimum Optimization (SMO) approach [9] can be used.

4.2 Wrapper Approaches for MKL

The main advantage of the wrapper approaches is that they are able to effectively exploit the off-the-shelf SVM solvers (e.g., LIBSVM). Below, we describe several representative wrapper approaches for MKL, including a Semi-Infinite Programming (SIP), a subgradient descent, a mirror-descend, and an extended level approach.

4.2.1 A Semi-infinite Programming Approach for MKL (MKL-SIP)

It was shown in [50] that the dual problem in (3) can be cast into the following SIP problem:

$$\begin{aligned} &\min_{\theta\in\mathbb{R},\boldsymbol{\beta}\in\Delta} \quad \theta \qquad\qquad (9)\\ &\text{s. t.} \quad \sum_{j=1}^{s}\beta_j\{\alpha^\top\mathbf{1}-\frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}_j(\boldsymbol{\alpha}\circ\mathbf{y})\}\geq\theta,\\ &\forall\alpha\in\mathcal{Q} \end{aligned}$$

When we use the domain Δ_1 for $\boldsymbol{\beta}$, the problem in (9) is reduced to a Semi-Infinite Linear Programming (SILP) problem. To solve (9), we first initialize the problem with a small number of linear constraints. We then solve (9) by alternating between two steps: (i) finding the optimal $\boldsymbol{\beta}$ and θ with fixed constraints, and (ii) finding the unsatisfied constraints with the largest violation under the fixed $\boldsymbol{\beta}$ and θ and adding them to the system. Note that in the second step, to find the most violated constraints, we need to solve the optimization problem

$$\max_{\alpha\in\mathcal{Q}} \quad \sum_{j=1}^{s}\beta_j S_j(\alpha) = \alpha^\top\mathbf{1}-\frac{1}{2}(\alpha\circ\mathbf{y})^\top K(\boldsymbol{\beta})(\alpha\circ\mathbf{y}),$$

a SVM problem using the combined kernel $\kappa(\cdot,\cdot;\boldsymbol{\beta})$ that can be solved using off-the-shelf SVM solver.

4.2.2 Subgradient Descent Approaches for MKL (MKL-SD & MKL-MD)

A popular wrapper approach for MKL is SimpleMKL [43], which solves the dual problem in (3) by a subgradient descent approach. The authors turn the convex concave optimization problem in (3) into a minimization problem $\min\limits_{\boldsymbol{\beta}\in\Delta} J(\boldsymbol{\beta})$, where the objective $J(\boldsymbol{\beta})$ is defined as

$$J(\boldsymbol{\beta}) = \max_{\boldsymbol{\alpha}\in\mathcal{Q}} -\frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}(\boldsymbol{\beta})(\boldsymbol{\alpha}\circ\mathbf{y})+\mathbf{1}^\top\boldsymbol{\alpha} \tag{10}$$

Since the partial gradient of $J(\boldsymbol{\beta})$ is given by $\partial_{\beta_j}J(\boldsymbol{\beta}) = 1-\frac{1}{2}(\boldsymbol{\alpha}^*\circ\mathbf{y})^\top\mathbf{K}_j(\boldsymbol{\alpha}^*\circ\mathbf{y}), j=1,\ldots,s,$

where $\boldsymbol{\alpha}^*$ is an optimal solution to (10), following the subgradient descent algorithm, we update the solution $\boldsymbol{\beta}$ by

$$\boldsymbol{\beta} \leftarrow \pi_\Delta \left(\boldsymbol{\beta} - \eta \partial J(\boldsymbol{\beta})\right)$$

where $\eta > 0$ is the step size determined by a line search [43] and $\pi_\Delta(\boldsymbol{\beta})$ projects $\boldsymbol{\beta}$ into the domain Δ. Similar approaches were proposed in [12], [39].

A generalization of the subgradient descent method for MKL is a mirror descent method (MKL-MD) [23]. Given a proximity function $w(\boldsymbol{\beta}', \boldsymbol{\beta})$, the current solution $\boldsymbol{\beta}^t$ and the subgradient $\partial J(\boldsymbol{\beta}^t)$, the new solution $\boldsymbol{\beta}^{t+1}$ is obtained by solving the following optimization problem

$$\boldsymbol{\beta}^{t+1} = \underset{\boldsymbol{\beta} \in \Delta}{\arg\min}\, \eta(\boldsymbol{\beta} - \boldsymbol{\beta}^t)^\top \partial J(\boldsymbol{\beta}^t) + w(\boldsymbol{\beta}^t, \boldsymbol{\beta}), \quad (11)$$

where $\eta > 0$ is the step size.

The main shortcoming of SimpleMKL arises from the high computational cost of line search. It was indicated in [8] that many iterations may be needed by the line search to determine the optimal step size. Since each iteration of the line search requires solving a kernel SVM, it becomes computationally expensive when the number of training examples is large. Another subtle issue of SimpleMKL, as pointed out in [43], is that it may not converge to the global optimum if the kernel SVM is not solved with high precision.

4.2.3 An Extended Level Method for MKL (MKL-Level)

An extended level method is proposed for L_1-MKL in [42]. To solve the optimization problem in (3), at each iteration, the level method first constructs a cutting plane model $g^t(\boldsymbol{\beta})$ that provides a lower bound for the objective function $J(\boldsymbol{\beta})$. Given $\{\boldsymbol{\beta}^j\}_{j=1}^t$, the solutions obtained for the first t iterations, a cutting plane model is constructed as $g^t(\beta) = \max_{1 \le j \le t} L(\boldsymbol{\beta}, \boldsymbol{\alpha}^j)$, where $\boldsymbol{\alpha}^j = \arg\max_{\boldsymbol{\alpha} \in Q} L(\boldsymbol{\beta}^j, \boldsymbol{\alpha})$. Given the cutting plane model, the level method then constructs a level set $\mathcal{S}_t$ as

$$\mathcal{S}_t = \{\boldsymbol{\beta} \in \Delta_1 : g^t(\boldsymbol{\beta}) \le l^t = \lambda \bar{L}^t + (1-\lambda)\underline{L}^t\}, \quad (12)$$

and obtain the new solution $\boldsymbol{\beta}^{t+1}$ by projecting $\boldsymbol{\beta}^t$ into $\mathcal{S}_t$, where $\bar{L}^t$ and $\underline{L}^t$, the upper and lower bounds for the optimal value $L(\boldsymbol{\beta}^*, \boldsymbol{\alpha}^*)$, are given by $\underline{L}^t = \min\limits_{\boldsymbol{\beta} \in \Delta} g^t(\boldsymbol{\beta})$ and $\bar{L}^t = \min\limits_{1 \le j \le t} L(\boldsymbol{\beta}^j, \boldsymbol{\alpha}^j)$.

Compared to the subgradient-based approaches, the main advantage of the extended level method is that it is able to exploit *all* the gradients computed in the past for generating new solutions, leading to a faster convergence to the optimal solution.

4.2.4 An Alternating Optimization Method for MKL (MKL-GL)

This approach was proposed in [43], [10] for L_1-MKL. It is based on the equivalence between group Lasso and MKL, and solves the following optimization problem for MKL

$$\min_{\substack{\boldsymbol{\beta} \in \Delta_1 \\ f_j \in \mathcal{H}_j}} \frac{1}{2}\sum_{j=1}^s \frac{\|f_j\|_{\mathcal{H}_j}^2}{\beta_j} + C\sum_{i=1}^n \ell\left(y_i \sum_{j=1}^s f_j(\mathbf{x}_i)\right) \quad (13)$$

To solve the optimization problem in (13), an alternating optimization method was proposed in in [43]. It alternates between two steps, i.e. the step of optimizing f_j under fixed $\boldsymbol{\beta}$ and the step of optimizing $\boldsymbol{\beta}$ given fixed f_j. The first step is equivalent to solving a kernel SVM with a combined kernel $\kappa(\cdot, \cdot; \beta)$, and the optimal solution in the second step is given by

$$\boldsymbol{\beta}_k = \frac{||f_k||_{\mathcal{H}_k}}{\sum_{k=1}^s ||f_k||_{\mathcal{H}_k}} \quad (14)$$

It was shown in [8] that the above approach can be extended to L_p-MKL.

4.3 Online Learning Algorithms for MKL

We briefly discuss online learning algorithms for MKL. Online learning is computationally efficient as it only needs to process one training example at each iteration. In [79], the authors propose several online learning algorithms for MKL that combine the Perceptron algorithm [80] with the Hedge algorithm [81]. More specifically, they apply the Perceptron algorithm to update the classifiers for the base kernels and the Hedge algorithm to learn the combination weights. In [21], Jie et al. present an online learning algorithm for MKL, based on the framework of the follow-the-regularized-leader (FTRL). One disadvantage of online learning for MKL is that it usually yields suboptimal recognition performance compared to the batch learning algorithms. As a result, we did not include online MKL in our empirical study.

4.4 Computational Efficiency

In this section we review the conflicting statements in MKL literature about the computational efficiency of different optimization algorithms for MKL. First, there is no consensus on the efficiency of the SIP based approach for MKL. While several studies show a slow convergence of the SIP based approach for MKL [48], [43], [42], [9], it was stated in [75] that only a few iterations would suffice when the number of relevant kernels is small. According to our empirical study, the SIP based approach can converge in a few iterations for L_p-MKL. On the other hand, SIP based approach takes many more iterations to converge for L_1-MKL.

Second, several studies have evaluated the training time of SimpleMKL in comparison to the other approaches for MKL, but with different conclusions. In [8] MKL-SIP is found to be significantly slower than SimpleMKL while the studies in [42], [41] report the opposite.

The main reason behind the conflicting conclusions is that the size of testbed (i.e. the number of training examples and the number of base kernels) varies significantly from one study to another (Table 2). When the number of kernels and the number of training examples are large, calculation and combination of the base kernels take a significant amount of the computational load, while for small datasets, the computational efficiency is mostly decided by the iteration complexity of algorithms. In addition, implementation details, including the choice of stopping criteria and programming tricks for calculating the combined kernel matrix, can also affect the running time.

In terms of visual object recognition, our empirical study shows that SimpleMKL is less efficient than MKL-SIP. Although SimpleMKL requires a smaller number of iterations, it takes significantly longer time to finish one iteration than the other approaches for MKL, due to the high computational cost of the line search. Overall, we observed that MKL-SIP is more efficient than the other wrapper optimization techniques for MKL whereas MKL-SMO is the fastest method for solving L_p-MKL.

5 Experiments

Our goal is to evaluate the classification performance of different MKL formulations and the efficiency of different optimization techniques for MKL. We focus on MKL algorithms for binary classification, and apply the one-versus-all (OvA) strategy to convert a multi-label learning problem into a set of binary classification problems, with one binary classification problem for each class. Among various formulations for MKL, we only evaluate algorithms for L_1 and L_p regularized MKL. As stated earlier, we do not consider (i) online MKL algorithms due to their suboptimal performance and (ii) nonlinear MKL algorithms due to their high computational costs.

The first objective of this empirical study is to compare L_1-MKL algorithms to the two simple baselines of kernel combination mentioned in Section 2.1, i.e., the single best performing kernel and the average kernel approach. As already mentioned in Section 2.1, there are contradictory statements from different studies regarding the comparison of MKL algorithms to these two baselines (Table 1). The goal of our empirical study is to examine and identify the factors that may contribute to the conflicting statements. The factors we consider here include (i) the number of training examples and (ii) the number of base kernels. The second objective of this study is to evaluate the classification performance of different MKL formulations for visual object recognition. In particular, we will compare L_1-MKL to L_p-MKL with $p = 2$ and $p = 4$. The final objective of this study is to evaluate the computational efficiency of different optimization algorithms for MKL. To this end, we choose six representative MKL algorithms in our study (See Section 5.2).

5.1 Datasets, Features and Kernels

Three benchmark datasets for object recognition are used in our study: Caltech 101 [5], Pascal VOC 2007 [82], and a subset of ImageNet [2]. All the experiments conducted in this study are repeated five times, each with an independent random partition of training and testing data. Average classification accuracies along with the associated standard deviation are reported.

Caltech 101 dataset has been used in many MKL studies. It is comprised of $9,146$ images from 101 object classes and an additional class of "background" images. Caltech 101 is a single-labeled dataset in which each image is assigned to one object class. To obtain the full spectrum of classification performance for MKL, we vary the number of training examples per class (10, 20, 30). We construct 48 base kernels (Table 3) for the Caltech 101

[2] www.image-net.org

dataset: 39 of them are built by following the procedure in [25] and the remaining 9 are constructed by following [49]. For all the features except the one that is based on geometric blur, RBF kernel with χ^2 distance is used as the kernel function [2]. For the geometric blur feature, RBF kernel with the average distance of the nearest descriptor pairs between two images is used [49].

TABLE 3: Description of the 48 kernels built for the Caltech-101 dataset.

Kernel indices	Description	Color Space	# levels for SPK
1-3	LBP [65]	Gray	3
4	LBP (combined histogram)	Gray	3
5-8	BOW with dense-SIFT (300 bins)	HSV	4
9-12	BOW with dense-SIFT (1000 bins)	Gray	4
13-16	BOW with dense-SIFT (1000 bins)	HSV	4
17-18	Mean over 100 subwindows [24]	Gray-HSV	1
19-22	BOW with dense-SIFT (300 bins)	Gray	4
23-26	Canny edge detector + histogram of unoriented gradient feature (40 bins)	Gray	4
27-30	Canny edge detector + histogram of oriented gradient feature (40 bins) [83]	Gray	4
31,34, 33,34	Product of kernels: {20 to 23}, {24 to 27}, {16 to 19}, and {4 to 7}		1
35	V1S+ feature [70]	Gray	1
36-38	Region covariance [84]	Gray	3
39	Product of kernels 4 to 7		1
40	Geometric blur [71]	Gray	1
41-43	BOW with dense-SIFT (300 bins)	Gray	4
44-46	BOW with dense-SIFT (300 bins)	HSV	4
47-48	BOW (300 visual words)[85] with self-similarity features	Gray	2

Pascal VOC 2007 dataset is comprised of $9,963$ images from 20 object classes. Unlike Caltech 101, more than half of the images in VOC 2007 are assigned to multiple classes. It is, a more challenging dataset than Caltech 101 because of the large variations in object size, orientation, and shape, as well as the occlusion problem. We vary the number of training examples, by randomly selecting 1%, 25%, 50%, and 75% of images to form the training set. Due to the different characteristics of the two datasets, we choose a different set of image features for VOC 2007. In particular, we follow [86] and create 15 sets of features: (i) GIST features [69]; (ii) six sets of color features generated by two different spatial pooling layouts [68] (1×1 and 3×1) and three types of color histograms (i.e. RGB, LAB, and HSV). (iii) eight sets of local features generated by two keypoint detection methods (i.e., dense sampling and Harris-Laplacian [63]), two spatial layouts (1×1 and 3×1), and two local descriptors (SIFT and robust hue [87]). A RBF kernel with χ^2 distance is applied to each of the 15 feature sets.

A Subset of ImageNet is used in [88] for evaluating object recognition. Following the protocol in [88], we include in this dataset about $81,738$ images from ImageNet that belong to 18 categories of VOC 2007. This is significantly larger than Caltech101 and VOC2007, making it possible to examine the scalability of MKL methods for object recognition. Both dense sampling and Harris-Laplacian [63] are used for keypoint detection, and SIFT is used as the local descriptor. We create four BoW models by setting the vocabulary size to be $1,000$ and applying two descriptor pooling techniques (i.e. max-pooling and mean-pooling) to a two level spatial pyramid kernel (i.e. 1×1 and 4×4 spatial partitionings). We also created six color histograms by applying two pooling techniques (i.e. max-pooling and mean-pooling) to three different color spaces, namely RGB, LAB and HSV. In total, ten kernels are created for the ImageNet dataset. We note that the number of base kernels we constructed for the ImageNet dataset is significantly smaller than the other two datasets because of the significantly larger number of images in the ImageNet dataset. The common practice for large scale datasets has been to use a small number of features/kernels for scalability concerns [88].

5.2 MKL Methods in Comparison

We divide the MKL methods of interest into two groups. The first group consists of the two simple baselines for kernel combination, i.e., the average kernel method (AVG) and the best performing kernel selected by the cross validation method (Single). The second group includes seven MKL methods designed for binary classification. These are: GMKL [12], SimpleMKL [43], VSKL [44], MKL-GL [8], MKL-Level [42], MKL-SIP [10], MKL-SMO [9]. The difference between the two subgradient descent based methods, SimpleMKL and GMKL, is that SimpleMKL performs a golden section search to find the optimal step size while GMKL applies a simple backtracking method.

In addition to different optimization algorithms, we have used both L_1-MKL and L_p-MKL with $p = 2$ and $p = 4$. For L_p-MKL, we apply MKL-GL, MKL-SIP, and MKL-SMO to solve the related optimization problems.

5.3 Implementation

To make a fair comparison, we follow [8] and implement [3] all wrapper MKL methods within the

[3] The code is available at http://www.cse.msu.edu/~bucakser/mkl.html

framework of SimpleMKL using MATLAB, where we used LIBSVM [89] as the SVM solver. For MKL-SIP and MKL-Level, CVX [90] and MOSEK [77] are used to solve the related optimization problems, as suggested in [42].

The same stopping criteria is applied to all baselines. The algorithms are stopped when one of the following criteria is satisfied: (i) the maximum number of iterations (specified as 40 for wrapper methods) is reached, (ii) the difference in the kernel coefficients $\boldsymbol{\beta}$ between two consecutive iterations is small (i.e., $||\boldsymbol{\beta}^t - \boldsymbol{\beta}^{t-1}||_\infty < 10^{-4}$), (iii) the duality gap drops below a threshold value (10^{-3}).

The regularization parameter C is chosen with a grid search over $\{10^{-2}, 10^{-1}, \ldots, 10^4\}$. The bandwidth of the RBF kernel is set to the average pairwise χ^2 distance of images.

In our empirical study, all the feature vectors are normalized to have the unit L_2 norm before they are used to construct the base kernels. According to [91] and [10], kernel normalization can have a significant impact on the performance of MKL. Various normalization methods have been proposed, including unit trace normalization [91], normalization with respect to the variance of kernel features [10], and spherical normalization [10]. However, since almost no difference was observed in the classification accuracy for all the methods in comparative study when applying the above techniques to normalize kernels, we will only report the performance without using kernel normalization.

The experiments with varied numbers of kernels on the ImageNet dataset were performed on a cluster of Sun Fire X4600 M2 nodes, each with 256 GB of RAM and 32 AMD Opteron cores. All other experiments were run on a different cluster, where each node has two four-core Intel Xeon E5620s at 2.4 GHz with 24 GB of RAM. We pre-compute all the kernel matrices and load the computed kernel matrices into the memory. This allows us to avoid re-computing and loading kernel matrices at each iteration of optimization.

5.4 Classification Performance of MKL

We evaluate the classification performance by the mean average precision (MAP) score. For a given class, each image is given a score by the classification method, and the images with scores higher than the threshold are returned as relevant images (i.e., images containing the objects from the given class). By varying the threshold, we obtain different sets of relevant images, each providing a value of precision and recall. The MAP score is computed based on the precision-recall pairs obtained by varying the threshold. For convenience, we report MAP score as a percentage number because the maximum value for MAP is 1.

5.4.1 Experiment 1: Classification Performance

Table 4 summarizes the classification results for the Caltech 101 dataset with 10, 20, and 30 training examples per class. First, we observe that both the MKL algorithms and the average kernel approach (AVG) outperform the best base kernel (Single). This is consistent with most of the previous studies [49], [6] Compared to the average kernel approach, we observe that the L_1-MKL algorithms have the worst performance when the number of training examples per class is small ($n = 10, 20$), but significantly outperform the average kernel approach when $n = 30$. This result explains the seemingly contradictory conclusions reported in the literature. When the number of training examples is insufficient to determine the appropriate kernel combination, it is better to assign all the base kernels equal weights. MKL becomes effective only when the number of training examples is large enough to determine the optimal kernel combination.

Next, we compare the performance of L_1-MKL to that of L_p-MKLs. We observe that L_1-MKL performs worse than L_p-MKLs ($p = 2, 4$) when the number of training examples is small (i.e., $n = 10, 20$), but outperforms L_p-MKLs when $n = 30$. This result again explains why conflicting results were observed in different MKL studies in the literature. Compared to L_p-MKL, L_1-MKL gives a sparser solution for the kernel combination weights, leading to the elimination of irrelevant kernels. When the number of training examples is small, it is difficult to determine the subset of kernels that are irrelevant to a given task. As a result, the sparse solution obtained by L_1-MKL may be inaccurate, leading to a relatively lower classification accuracy than L_p-MKL. L_1-MKL becomes advantageous when the number of training examples is large enough to determine the subset of relevant kernels.

We observe that there is no significant difference in the classification performance between different MKL optimization techniques. This is not surprising since they solve the same optimization problem. It is however interesting to note that although different optimization algorithms converge to the

TABLE 4: Classification results (MAP) for the Caltech 101 dataset. We report the average values over five random splits and the associated standard deviation.

Baseline	Norm	Number of training instances per class 10	20	30
Single		45.3 ± 0.9	55.2 ± 0.9	70.6 ± 0.9
Average		**59.0 ± 0.7**	**69.7 ± 0.6**	77.2 ± 0.5
GMKL	$p = 1$	54.2 ± 1.1	64.1 ± 0.7	**84.8 ± 0.7**
SimpleMKL	$p = 1$	53.6 ± 0.9	63.4 ± 0.6	**84.6 ± 0.5**
VSKL	$p = 1$	53.9 ± 0.9	64.0± 0.6	**85.3 ± 0.5**
level-MKL	$p = 1$	54.7 ± 1.0	63.4 ± 0.6	**84.4** ± 0.4
MKL-GL	$p = 1$	54.3 ± 1.0	64.7 ± 0.7	**85.4 ± 0.4**
MKL-GL	$p = 2$	**60.3 ± 0.6**	**70.7 ± 1.0**	80.0 ± 0.6
MKL-GL	$p = 4$	**60.1 ± 0.7**	**70.7 ± 1.0**	80.0 ± 0.6
MKL-SIP	$p = 1$	53.8 ± 0.6	63.8 ± 0.9	**83.9 ± 0.7**
MKL-SIP	$p = 2$	**60.1 ± 0.6**	**70.7 ± 1.0**	79.1 ± 0.6
MKL-SIP	$p = 4$	**59.4 ± 0.6**	**70.0 ± 1.0**	77.5 ± 0.5
MKL-SMO	$p = 2$	**59.8 ± 0.5**	**69.7.0 ± 0.9**	79.3 ± 0.9
MKL-SMO	$p = 4$	**59.6 ± 0.4**	**69.6 ± 0.7**	79.0 ± 0.5

TABLE 5: Classification results (MAP) for the VOC 2007 dataset. We report the average values over five random splits and the associated standard deviation.

baseline	Percentage of the samples used for training 1%	25%	50%	75%
Single	**23.4 ± 0.1**	44.7 ± 0.8	48.6 ± 0.8	50.0 ± 0.8
Average	21.9 ± 0.5	48.2 ± 0.8	54.5 ± 0.8	57.5 ± 0.8
L_1-MKL	**23.5 ± 0.7**	**51.9 ± 0.4**	**57.4 ± 0.4**	**59.9 ± 0.9**
L_2-MKL	22.7 ± 0.4	49.8 ± 0.2	**57.3 ± 0.2**	**60.6± 0.5**

TABLE 6: Comparison with the state-of-the-art performance for object classification on Caltech 101 (measured by classification accuracy) and VOC 2007 datasets (measured by MAP).

Caltech 101 (30 per class)			
This paper		state-of-the-art	
AVG :	77.09	**[6]**:	**84.3**
L_1-MKL :	79.93	[92]:	81.9
L_2-MKL :	77.94	[93]:	80.0
VOC 2007			
This paper		state-of-the-art	
AVG:	55.4	**[94]**:	**73.0**
L_1-MKL:	57.2	[95]:	63.5
L_2-MKL:	57.4	[96]:	61.7

same solution, they could behave very differently over iterations. In Fig. 3, we show how the classification performances of the L_1-MKL algorithms change over the iterations for three classes from Caltech101 dataset. We observe that

- SimpleMKL converges in a smaller number of iterations than the other L_1-MKL algorithms. Note that convergence in a smaller number of iterations does not necessarily mean a shorter training time, as SimpleMKL takes significantly longer time to finish one iteration than the other algorithms.
- The classification performance of MKL-SIP fluctuates significantly over iterations. This is due to the greedy nature of MKL-SIP as it selected the most violated constraints at each iteration of optimization.

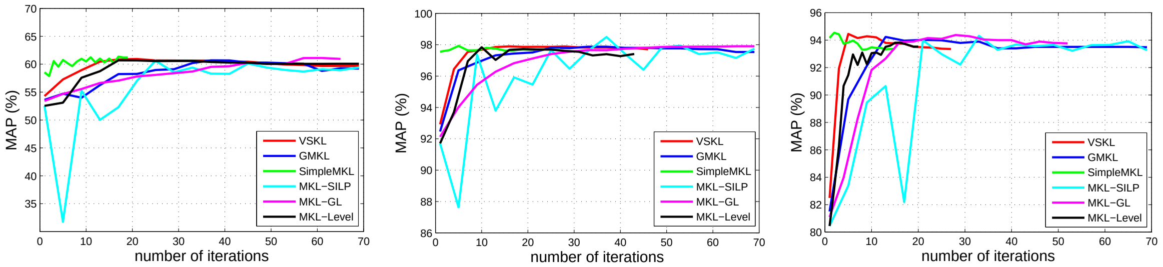
Due to the space constraint, from now on, unless specified, we will only report the results of one representative method for both L_1-MKL (Level-MKL) and L_p-MKL (MKL-SIP, $p = 2$).

Table 5 shows the classification results for VOC 2007 dataset with 1%, 25%, 50%, and 75% of images used for training. These results confirm the conclusions drawn from Caltech 101 dataset: MKL methods do not outperform the simple baseline (i.e. the best single kernel) when the number of training examples is small (i.e., 1%); the advantage of MKL is obvious only when the number of training examples is sufficiently large.

Finally, we compare in Table 6 the performance of MKL to that of the state-of-the-art methods for object recognition on the Caltech 101 and VOC 2007 datasets. For Caltech 101, we use the standard splitting formed by randomly selecting 30 training examples for each class, and for VOC 2007, we use the default partitioning. We observe that the L_1-MKL achieves similar classification performance as the state-of-the-art approaches for the Caltech 101 dataset. However, for the VOC 2007 dataset, the performance of MKL is significantly worse than the best ones [94], [95]. The gap in the classification performance is because object detection (localization) methods are utilized in [94], [95] to boost the recognition accuracy for the VOC 2007 dataset but not in this paper. We also note that the authors of [96] get a better result by using only one strong and well-designed (Fisher vector) representation compared to the MKL results we report. Interested readers are referred to [96], which provides an excellent empirical study on how the different steps of the BoW model can affect the classification results. Note that the performance of MKL techniques can be improved further by using the different and stronger options discussed in [96].

5.4.2 *Experiment 2: Number of Kernels vs Classification Accuracy*

In this experiment, we examine the performance of MKL methods with increasing numbers of base kernels. To this end, we rank the kernels in the descending order of their weights computed by L_1-



(a) Caltech 101 - class 3 (b) Caltech 101 - class 10 (c) Caltech 101 - class 15

Fig. 3: Mean average precision (MAP) scores of different L_1-MKL methods vs number of iterations.

MKL, and measure the performance of MKL and baseline methods by adding kernels sequentially from the ranking list. The number of kernels is varied from 2 to 48 for the Caltech 101 dataset and from 2 to 15 for VOC 2007 dataset. Figs. 4 and 5 summarizes the classification performance of MKL and baseline methods as the number of kernels is increased. We observe that when the number of kernels is small, all the methods are able to improve their classification performance with increasing number of kernels. But, the performance of average kernel and L_2-MKL starts to drop as more and more weak kernels (i.e. kernels with small weights computed by L_1-MKL) are added. In contrast, we observe a performance saturation for L_1-MKL after five to ten kernels have been added. We thus conclude that L_1-MKL is more resilient to the introduction of weak kernels than the other kernel combination methods.

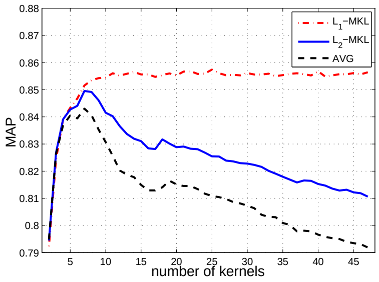


Fig. 4: The change in MAP score with respect to the number of base kernels for the Caltech 101 dataset.

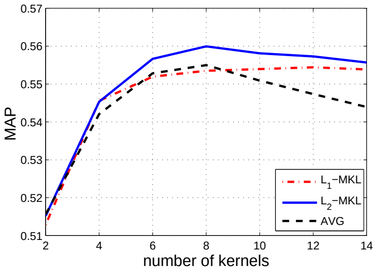


Fig. 5: The change in MAP score with respect to the number of base kernels for the VOC 2007 dataset.

5.5 Computational Efficiency

To evaluate the learning efficiency of MKL algorithms, we report the training time for the simulations with different numbers of training examples and base kernels. Many studies on the computational efficiency of MKL algorithms focused on the convergence rate (i.e. number of iterations) [42], which is not necessarily the deciding factor in determining the training time. For instance, according to Fig. 3, although SimpleMKL requires a smaller number of iterations to obtain the optimal solution than the other L_1-MKL approaches, it is significantly slower in terms of running time than the other algorithms because of its high computational cost per iteration. Thus, besides the training time, we also examine the sparseness of the kernel coefficients, which can significantly affect the efficiency of both training and testing.

5.5.1 Experiment 4: Evaluation of Training Time

We first examine how the number of training examples affect the training time of the wrapper methods. Table 7 summarizes the training time of different MKL algorithms for Caltech 101 dataset and VOC 2007 datasets. We also include in the table the number of iterations and the time for computing the combined kernel matrices. We did not include the time for computing kernel matrices because it is shared by all the methods. We draw the following observations from Table 7:

- The L_p-MKL methods require a considerably smaller number of iterations than the L_1-MKL methods, indicating they are computationally more efficient. This is not surprising because L_p-MKL employs a smooth objective function that leads to more efficient optimization [73].
- Since a majority of the training times is spent on computing combined kernel matrices, the time difference between different L_1-MKL methods is mainly due to the sparseness of their intermediate solutions. Since MKL-SIP yields sparse solutions throughout its optimization process, it is the most efficient wrapper algorithm for MKL. Although SimpleMKL converges in a smaller number of iterations than the other L_1-MKL methods, it is not as efficient as the MKL-SIP method because it does not generate sparse intermediate solutions.

In the second set of experiments, we evaluate the training time as a function of the number of base kernels. For both Caltech 101 and VOC 2007 datasets, we choose 15 kernels with the best classification accuracy, and create 15, 30, and 60 kernels by simply varying the kernel bandwidth (i.e., from 1 times, to 1.5 and 2 times the average χ^2 distance). The number of training examples is set to be 30 per class for Caltech 101 and 50% of images are used for training for VOC 2007. Table 8 summarizes for different MKL algorithms, the training time, the number of iterations, and the time for computing the combined kernel matrices. Overall, we observe that L_p-MKL is still more efficient than L_1-MKL, even when the number of base kernels is large. But the gap in the training time between L_1-MKL and L_p-MKL becomes significantly smaller for the MKL-SIP method when the number of combined kernels is large. In fact, for the Caltech 101 dataset with 108 base kernels, MKL-SIP for L_1-MKL is significantly more efficient than MKL-SIP for L_p-MKL ($p > 1$). This is because of the sparse solution obtained by MKL-SIP for L_1-MKL, which leads to less time on computing the combined kernels than MKL-SIP for L_p-MKL, as indicated in Table 8.

As discussed in Section 5.3, we cannot compare MKL-SMO directly with the other baselines in terms of training times since they are not coded in the same platform. Instead, we use the code provided by the authors of MKL-SMO [9] to compare it to the C++ implementation of MKL-SIP, the fastest wrapper approach, which is available within the Shogun package [97]. We fix $p = 2$, vary the number of training samples for a fixed number of kernels (48 for Caltech 101 and 15 for VOC 2007) and the number of base kernels for a fixed number of samples (2040 for Caltech 101 and 5011 for VOC 2007). Table 9 shows that MKL-SMO is significantly faster than MKL-SIP on both datasets, demonstrating the advantage of a well-designed direct MKL optimization method against the wrapper approaches for L_p-MKL. We finally note that MKL-SMO cannot be applied to L_1-MKL which often demonstrates better performance with a modest number of training examples.

TABLE 9: Comparison of training time between MKL-SMO and MKL-SIP

		Number of training samples		
Caltech 101		$\underline{n = 10}$	$\underline{n = 20}$	$\underline{n = 30}$
	MKL-SIP	3.6 ±0.2	6.5± 0.3	11.8 ± 0.7
	MKL-SMO	**0.2 ±0.1**	**2.3 ± 0.2**	**3.8 ± 0.5**
VOC 2007		$\underline{25\%}$	$\underline{50\%}$	$\underline{75\%}$
	MKL-SIP	15.5 ± 1.6	145.6 ± 3.9	360.7 ± 8.4
	MKL-SMO	**3.5 ± 0.7**	**14.2± 1.8**	**33.1± 3.0**
		Number of base kernels		
Caltech 101		$\underline{K = 48}$	$\underline{K = 63}$	$\underline{K = 108}$
	MKL-SIP	6.5 ± 0.3	13.6 ± 2.9	19.8 ± 3.4
	MKL-SMO	**2.3 ± 0.2**	**3.2± 0.8**	**6.3± 1.0**
VOC 2007		$\underline{K = 15}$	$\underline{K = 30}$	$\underline{K = 75}$
	MKL-SIP	145.6 ± 3.9	542.0 ± 32.8	1412.1 ± 63.4
	MKL-SMO	**14.2 ± 1.8**	**29.1± 2.8**	**77.8± 10.3**

5.5.2 *Experiment 5: Evaluation of Sparseness*

We evaluate the sparseness of MKL algorithms by examining the sparsity of the solution for kernel combination coefficients. In Figs. 6 and 7, we show how the size of active kernel set (i.e., kernels with non-zero combination weights) changes over the iterations for MKL-SIP with three types of regularizers: L_1-MKL, L_2-MKL and L_4-MKL. Note that it is difficult to distinguish the results of L_2-MKL and L_4-MKL from each other as they are identical.

As expected, L_1-MKL method produces significantly sparser solutions than L_p-MKL. As a result, although L_p-MKL is more efficient for training because it takes a smaller number of iterations to train L_p-MKL than L_1-MKL, we expect L_1-MKL to be computationally more efficient for testing than L_p-MKL as most of the base kernels are eliminated and need not to be considered.

5.6 Large-scale MKL on ImageNet

To evaluate the scalability of MKL, we perform experiments on the subset of ImageNet consisting

TABLE 7: Total training time (secs), number of iterations, and total time spent on combining the base kernels (secs) for different MKL algorithms vs number of training examples for the Caltech 101(left) and VOC 2007 datasets (right).

	Caltech 101 dataset		
	10 training instances per class		
baseline	training	#iter	KerComb
GMKL-L_1	34.6 ± 8.6	38.4 ± 2.0	27.9 ± 7.7
SimpleMKL-L_1	55.7 ± 25.3	17.2 ± 6.8	46.1 ± 22.0
VSKL-L_1	14.1 ± 2.3	38.3 ± 4.3	11.1 ± 1.7
MKL-GL-L_1	21.9 ± 0.8	40.0 ± 0.0	19.5 ± 0.8
MKL-GL-L_2	5.3 ± 0.6	8.8 ± 1.0	4.8 ± 0.6
MKL-GL-L_4	**3.5 ± 0.2**	**5.9 ± 0.4**	3.2 ± 0.2
MKL-Level-L_1	8.0 ± 2.3	33.0 ± 9.5	5.5 ± 1.4
MKL-SIP-L_1	5.4 ± 0.9	39.4 ± 2.6	**2.1 ± 0.3**
MKL-SIP-L_2	**3.8±1.2**	**5.6±0.9**	**2.4±1.1**
MKL-SIP-L_4	**3.3±0.6**	**4.4±0.5**	**1.8±0.6**
	30 training instances per class		
baseline	training	#iter	KerComb
GMKL-L_1	256.7 ± 47.7	38.6 ± 1.8	212.5 ± 42.3
SimpleMKL-L_1	585.6 ± 204.7	19.0 ± 7.5	494.4 ± 174.7
VSKL-L_1	121.9 ± 22.4	36.6 ± 5.1	103.5 ± 17.7
MKL-GL-L_1	197.1 ± 9.1	39.8 ± 1.0	178.3 ± 8.5
MKL-GL-L_2	50.8 ± 5.6	9.3 ± 1.0	46.3 ± 5.2
MKL-GL-L_4	32.5 ± 1.6	5.9 ± 0.3	29.6 ± 1.5
MKL-Level-L_1	63.3 ± 22.1	27.5 ± 11.1	47.9 ± 14.9
MKL-SIP-L_1	44.3 ± 6.1	39.7 ± 2.9	23.2 ± 2.7
MKL-SIP-L_2	30.4±4.2	6.3±1.0	25.2±3.9
MKL-SIP-L_4	**22.6±2.6**	**4.7±0.5**	**18.2±2.1**

	VOC 2007 dataset		
	2500 training instances		
baseline	training	#iter	KerComb
GMKL-L_1	117.6 ± 16.3	39.0 ± 0.0	67.4 ± 7.7
SimpleMKL-L_1	175.1 ± 77.4	16.7 ± 7.3	112.9 ± 48.3
VSKL-L_1	45.2 ± 6.1	37.0 ± 3.4	25.3 ± 2.2
MKL-GL-L_1	62.6 ± 4.7	40.0 ± 0.0	43.5 ± 0.6
MKL-GL-L_2	14.5 ± 1.3	9.3 ± 0.6	10.2 ± 0.7
MKL-GL-L_4	**8.0 ± 0.8**	5.2 ± 0.4	5.6 ± 0.5
MKL-Level-L_1	40.1 ± 10.8	35.0 ± 7.7	20.2 ± 4.0
MKL-SIP-L_1	34.6 ± 6.8	39.9 ± 0.5	12.7 ± 1.4
MKL-SIP-L_2	**9.6±1.9**	5.7±0.5	4.9±0.4
MKL-SIP-L_4	**7.1±1.1**	**4.0±0.0**	**3.5±0.1**
	7500 training instances		
baseline	training	#iter	KerComb
GMKL-L_1	1133.2 ± 252.8	39.0 ± 0.0	646.9 ± 98.2
SimpleMKL-L_1	1671.3 ± 919.1	16.8 ± 6.4	1019.7 ± 424.8
VSKL-L_1	330.0 ± 49.2	29.9 ± 3.8	190.9 ± 22.8
MKL-GL-L_1	549.2 ± 79.8	40.0 ± 0.0	373.8 ± 4.2
MKL-GL-L_2	130.1 ± 17.7	9.5 ± 0.5	89.4 ± 6.1
MKL-GL-L_4	**74.9 ± 11.1**	5.3 ± 0.5	51.2 ± 4.5
MKL-Level-L_1	297.3 ± 95.2	31.1 ± 8.1	151.9 ± 31.0
MKL-SIP-L_1	309.0 ± 94.5	40.0 ± 0.0	117.0 ± 6.4
MKL-SIP-L_2	**84.3±24.5**	6.1±0.3	47.3±3.0
MKL-SIP-L_4	**56.4±14.7**	**4.1±0.3**	**31.5±2.2**

TABLE 8: Total training time (secs), number of iterations, and total time spent on combining the base kernels (secs) for different MKL algorithms vs number of base kernels for the Caltech 101 (left) dataset and VOC 2007 (right) datasets.

	Caltech 101 dataset		
	63 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	718.1 ± 169.8	38.8 ± 0.8	625.3 ± 152.9
SimpleMKL-L_1	1255.2 ± 350.9	17.3 ± 6.5	1047.6 ± 285.8
VSKL-L_1	398.1 ± 123.7	36.3 ± 5.2	345.6 ± 101.5
MKL-GL-L_1	397.1 ± 30.0	39.8 ± 1.0	351.9 ± 26.7
MKL-GL-L_2	118.8 ± 14.7	9.3 ± 1.0	108.5 ± 13.7
MKL-GL-L_4	**84.6 ± 5.8**	**6.0 ± 0.0**	**77.3 ± 4.8**
MKL-Level-L_1	204.1 ± 75.7	27.8 ± 10.4	167.2 ± 56.1
MKL-SIP-L_1	147.8 ± 29.8	39.8 ± 2.4	**85.3 ± 15.0**
MKL-SIP-L_2	**114.7±36.7**	7.9±0.7	**102.7±33.6**
MKL-SIP-L_4	**111.1±38.8**	7.5±0.8	**98.3±34.5** 9
	108 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	1170.5 ± 208.7	38.9 ± 0.8	1049.2 ± 190.7
SimpleMKL-L_1	2206.3 ± 580.1	17.2 ± 6.4	1960.3 ± 503.5
VSKL-L_1	569.9 ± 160.3	35.6 ± 5.9	491.8 ± 131.2
MKL-GL-L_1	604.6 ± 69.9	39.6 ± 1.6	546.6 ± 66.0
MKL-GL-L_2	226.3 ± 24.8	9.5 ± 1.0	212.0 ± 23.6
MKL-GL-L_4	**169.1 ± 16.0**	6.0 ± 0.1	**158.2 ± 14.5**
MKL-Level-L_1	405.8 ± 152.7	29.5 ± 9.5	343.7 ± 121.3
MKL-SIP-L_1	**192.1 ± 41.3**	39.9 ± 0.9	110.1 ± 18.1
MKL-SIP-L_2	634.1±107.2	6.8±1.3	582.1±106.3
MKL-SIP-L_4	407.2±80.2	**4.6±0.6**	368.4±67.9

	VOC 2007 dataset		
	30 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	1816.8 ± 405.8	37.8 ± 5.4	1186.9 ± 270.4
SimpleMKL-L_1	2335.3 ± 991.9	11.2 ± 7.1	1581.6 ± 626.4
VSKL-L_1	880.2 ± 128.5	30.6 ± 3.8	525.5 ± 75.3
MKL-GL-L_1	853.5 ± 206.1	40.0 ± 0.0	561.8 ± 107.3
MKL-GL-L_2	282.4 ± 64.2	9.6 ± 0.5	218.2 ± 46.3
MKL-GL-L_4	**190.1 ± 23.9**	**6.0 ± 0.0**	147.4 ± 11.0
MKL-Level-L_1	665.4 ± 114.7	36.8 ± 5.1	404.7 ± 40.2
MKL-SIP-L_1	460.0 ± 135.5	40.0 ± 0.0	170.6 ± 23.1
MKL-SIP-L_2	**240.8±62.5**	8.7±1.6	**154.5±43.5**
MKL-SIP-L_4	**170.1±16.5**	**6.2±0.4**	**115.1±15.4**
	75 base kernels		
baseline	training	#iter	KerComb
GMKL-L_1	3975.3 ± 890.0	34.2 ± 8.8	3072.5 ± 724.5
SimpleMKL-L_1	3416.3 ± 1299.7	**8.3 ± 7.8**	2776.4 ± 885.7
VSKL-L_1	1587.9 ± 238.8	29.4 ± 3.7	909.3 ± 122.2
MKL-GL-L_1	1500.4 ± 239.4	40.0 ± 0.0	1043.8 ± 87.6
MKL-GL-L_2	629.5 ± 84.0	9.8 ± 0.4	520.4 ± 47.7
MKL-GL-L_4	**346.2 ± 45.3**	6.0 ± 0.0	**286.2 ± 31.9**
MKL-Level-L_1	1136.8 ± 328.9	36.7 ± 3.1	702.2 ± 177.7
MKL-SIP-L_1	686.8 ± 262.9	40.0 ± 0.0	**228.5 ± 46.0**
MKL-SIP-L_2	**413.9±258.1**	**3.8±1.7**	**302.2±135.7**
MKL-SIP-L_4	**566.4±141.9**	**5.0±0**	424.2±81.5

of over $80,000$ images. Fig. 9 shows the classification performance of MKL and baseline methods with the number of training images per class varied in powers of 2 ($2^1, 2^2, ..., 2^{11}$). Similar to the experimental results for Caltech 101 and VOC 2007, we observed that the difference between L_1-MKL and the average kernel method is significant only when the number of training examples per class is sufficiently large (i.e. ≥ 16). We also observed that the difference between L_1-MKL and the average kernel method starts to diminish when the number of training examples is increased over 256 per class. We believe that the diminishing gap between MKL and the average kernel method with increasing number of training examples can be attributed to the fact that all the 10 kernels constructed for the ImageNet dataset are *strong* kernels and provide informative features for object recognition. This is reflected in the kernel combination weights learned by the MKL method: most of the base kernels

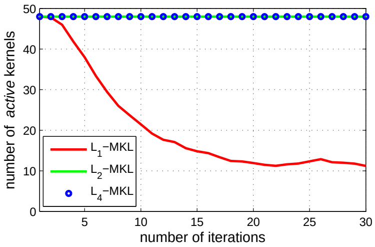


Fig. 6: Number of active kernels learned by the MKL-SIP algorithm vs number of iterations for the Caltech 101 dataset.

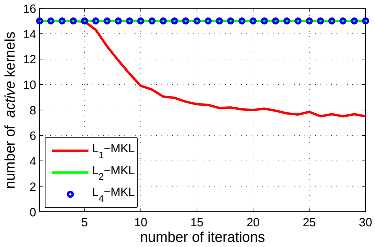


Fig. 7: Number of active kernels learned by the MKL-SIP algorithm vs number of iterations for the VOC 2007 dataset.

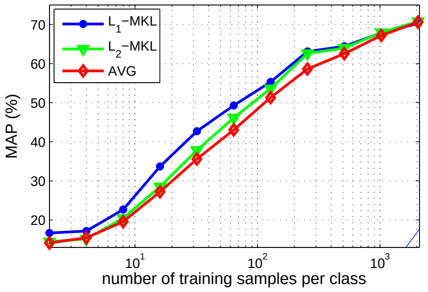


Fig. 8: Classification performance for different training set sizes for the ImageNet dataset.

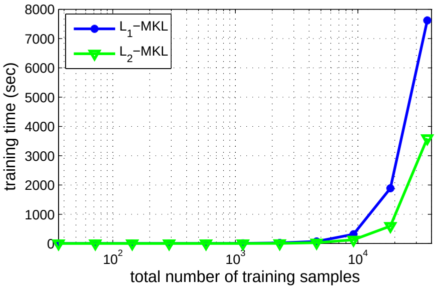


Fig. 9: Training times for L_1-MKL and L_2-MKL on different training set sizes for the ImageNet dataset.

received significant non-zero weights.

Fig. 9 shows the running time of MKL with a varied number of training examples. Similar to the experimental results for Caltech 101 and VOC 2007, we observe that L_2-MKL is significantly more efficient than L_1-MKL. We also observe that the running time for both L_1-MKL and L_2-MKL increases almost quadratically in the size of training data, making it difficult to scale to millions of training examples. We thus conclude that although MKL is effective in combining multiple image representations for object recognition, scalability of MKL algorithms is an open problem.

6 Summary and Conclusions

We have reviewed different formulations of multiple kernel learning and related optimization algorithms, with an emphasis on the application to visual object recognition. We highlighted the conflicting conclusions drawn by published studies on the empirical performance of different MKL algorithms. We have attempted to resolve these inconsistent conclusions by addressing the experimental setups in the published studies. Through our extensive experiments on two standard datasets used for visual object recognition, we are able to make the following conclusions:

- Overall, MKL is significantly more effective than the simple baseline for kernel combination (i.e., selecting the best kernel by cross validation or taking the average of multiple kernels), particularly when there are a large number of base kernels available and the number of training examples is sufficiently large. However, MKL is not recommended for object recognition when the base kernels are strong, and the number of training examples are sufficient enough to learn a reliable prediction for each base kernel.
- Compared to L_p-MKL, L_1-MKL is overall more effective for object recognition and is significantly more robust to the weaker kernels with low classification performance.
- MKL-SMO, which is not a wrapper method but a direct optimization technique, is the fastest MKL baseline. However, it does not address the L_1-MKL formulation.
- Among various algorithms proposed for L_1-MKL, MKL-SIP is overall the most efficient for object recognition, because it produces sparse

intermediate solutions throughout the optimization process.

- L_p-MKL is significantly more efficient than L_1-MKL because it converges in a significantly smaller number of iterations. But, neither L_1-MKL nor L_p-MKL scales well to very large datasets.
- L_1-MKL can be more efficient than L_p-MKL in terms of testing time. This is because L_1-MKL generates sparse solutions and, therefore, will only use a small portion of the base kernels for prediction.

In summary, we conclude that MKL is an extremely useful tool for visual object recognition because it provides a principled way to combine the strengths of different object representations. Although it is important to further improve the classification accuracy of MKL, it is much more critical to improve the overall computational efficiency of MKL. The existing algorithms for MKL do not scale to large datasets with millions of images.

Acknowledgments

The authors would like to thank Zenglin Xu and Peter Gehler for sharing their codes and data. This work was supported in part by National Science Foundation (IIS-0643494), US Army Research (ARO Award W911NF-08-010403) and Office of Naval Research (ONR N00014-09-1-0663). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NFS, ARO, and ONR. Part of Anil Jain's research was supported by WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-10008).

References

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

[2] J. Zhang, S. Lazebnik, and C. Schmid, "Local features and Kernels for Classification of Texture and Object Categories: a Comprehensive Study," *International Journal of Computer Vision*, vol. 73, pp. 213–238, 2007.

[3] Q. Chen, Z. Song, S. Liu, X. Chen, X. Yuan, T.-S. Chua, S. Yan, Y. Hua, Z. Huang, and S. Shen, "Boosting Classification with Exclusive Context," in *PASCAL Visual Object Classes Challenge Workshop*, 2010.

[4] M. A. Tahir, K. van de Sande, J. Uijlings, F. Yan, X. Li, K. Mikolajczyk, J. Kittler, T. Gevers, and A. Smeulders, "SurreyUVA_SRKDA method," in *PASCAL Visual Object Classes Challenge Workshop*, 2008.

[5] L. Fei-fei, R. Fergus, S. Member, and P. Perona, "One-shot Learning of Object Categories," *IEEE PAMI*, vol. 28(4), pp. 594 – 611, 2006.

[6] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao, "Group-Sensitive Multiple Kernel Learning for Object Categorization," in *ICCV*, 2009.

[7] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple Kernels for Object Detection," in *ICCV*, 2009.

[8] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and Efficient Multiple Kernel Learning by Group Lasso," in *ICML*, 2010.

[9] S. Vishwanathan, Z. Sun, N. Ampornpunt, and M. Varma, "Multiple Kernel Learning and the SMO Algorithm," in *NIPS 23*, 2010, pp. 2361–2369.

[10] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm Multiple Kernel Learning," *Journal of Machine Learning Research*, vol. 12, pp. 953–997, 2011.

[11] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien, "Efficient and Accurate Lp-Norm Multiple Kernel Learning," in *NIPS 22*, 2009, pp. 997–1005.

[12] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *ICML*, June 2009, pp. 1065–1072.

[13] F. Yan, K. Mikolajczyk, M. Barnard, H. Cai, and J. Kittler, "Lp Norm Multiple Kernel Fisher Discriminant Analysis for Object and Image Categorisation," in *CVPR*, 2010, pp. 3626–3632.

[14] T. Damoulas and M. A. Girolami, "Probabilistic Multi-class Multi-kernel Learning," *Bioinformatics*, vol. 24, pp. 1264–1270, 2008.

[15] C. Longworth and M. J. Gales, "Multiple Kernel Learning for Speaker Verification," in *ICASSP*, 2008, pp. 1581–1584.

[16] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple Kernel Learning, Conic Duality, and the SMO Algorithm," in *ICML*, 2004.

[17] D. P. Lewis, T. Jebara, and W. S. Noble, "Nonstationary Kernel Combination," in *ICML*, 2006, pp. 553–560.

[18] P. V. G. Person and S. Nowozin, "Infinite Kernel Learning," in *NIPS 2008 Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[19] Z. Wang, S. Chen, and T. Sun, "MultiK-MHKS: A Novel Multiple Kernel Learning Algorithm," *IEEE PAMI*, vol. 30, pp. 348–353, 2008.

[20] H. Do, A. Kalousis, A. Woznica, and M. Hilario, "Margin and Radius Based Multiple Kernel Learning," in *ECML PKDD*, 2009, pp. 330–343.

[21] L. Jie, F. Orabona, M. Fornoni, B. Caputo, and N. Cesa-bianchi, "OM-2: An online multi-class Multi-Kernel Learning Algorithm," in *IEEE Online Learning for Computer Vision Workshop*, 2010, pp. 43–50.

[22] S. Bucak, R. Jin, and A. Jain, "Multi-label Multiple Kernel Learning by Stochastic Approximation: Application to Visual Object Recognition," in *NIPS 23*, 2010, pp. 325–333.

[23] J. Saketha Nath, G. Dinesh, S. Raman, C. Bhattacharyya, A. Ben-Tal, and K. Ramakrishan, "On the Algorithmics and Applications of a Mixed-norm Based Kernel Learning Formulation," in *NIPS 22*, 2009.

[24] P. V. Gehler and S. Nowozin, "Let the Kernel Figure it Out: Principled Learning of Pre-processing for Kernel Classifiers," in *ICCV*, 2009.

[25] P. V. Gehler and S.Nowozin, "On Feature Combination for Multiclass Object Classification," in *ICML*, 2009.

[26] S. Nakajima, A. Binder, C. Mller, W. Wojcikiewicz, M. Kloft, U. Brefeld, K.-R. Mller, and M. Kawanabe, "Multiple Kernel Learning for Object Classification," in *the 12th Workshop on Information-based Induction Sciences*, 2009.

[27] J. Ren, Z. Liang, and S. Hu, "Multiple kernel learning

improved by mmd," in *International Conference on Advanced Data Mining and Applications*, 2010, pp. 63–74.

[28] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning Nonlinear Combinations of Kernels," in *NIPS 22*, 2009, pp. 396–404.

[29] ——, "L_2 Regularization for Learning Kernels," in *Conference on Uncertainty in Artificial Intelligence*, 2009.

[30] Z. Xu, R. Jin, S. Zhu, M. Lyu, and I. King, "Smooth Optimization for Effective Multiple Kernel Learning," in *AAAI Conference on Artificial Intelligence*, 2010.

[31] R. Tomioka and T. Suzuki, "Sparsity-accuracy Trade-off in MKL," in *NIPS Workshop on Understanding Multiple Kernel Learning Methods*, 2009.

[32] M. Gnen and E. Alpaydin, "Multiple Kernel Learning Algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[33] G. Lanckriet, N. Cristianini, P. Bartlett, and L. E. Ghaoui, "Learning the Kernel Matrix with Semi-Definite Programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[34] S. Sonnenburg, G. Rätsch, and C. Schäfer, "A General and Efficient Multiple Kernel Learning Algorithm," in *NIPS 18*, 2006, pp. 1273–1280.

[35] M. Kowalski, M. Szafranski, and L. Ralaivola, "Multiple Indefinite Kernel Learning with Mixed Norm Regularization," in *ICML*, 2009, pp. 545–552.

[36] C. Cortes, M. Mohri, and A. Rostamizadeh, "Generalization Bounds for Learning Kernels," in *ICML*, 2010.

[37] Z. Hussain and J. Shawe-Taylor, "A Note on Improved Loss Bounds for Multiple Kernel Learning," Tech. Rep., 2011.

[38] M. Kloft, U. Rückert, and P. L. Bartlett, "A Unifying View of Multiple Kernel Learning," in *ECML PKDD*, 2010, pp. 66–81.

[39] K. Gai, G. Chen, and C. Zhang, "Learning Kernels with Radiuses of Minimum Enclosing Balls," in *NIPS 23*, 2010, pp. 649–657.

[40] F. Yan, K. Mikolajczyk, J. Kittler, and A. Tahir, "A Comparison of L1 Norm and L2 Norm Multiple Kernel SVMs in Image and Video Classification," in *International Workshop on Content-Based Multimedia Indexing*, 2009.

[41] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More Efficiency in Multiple Kernel Learning," in *ICML*, 2007.

[42] Z. Xu, R. Jin, I. King, and M. R. Lyu, "An Extended Level Method for Efficient Multiple Kernel Learning," in *NIPS 21*, 2009, pp. 1825–1832.

[43] A. Rakotomamonjy, F. Bach, Y. Grandvalet, and S. Canu, "SimpleMKL," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[44] J. Aflalo, A. Ben-Tal, C. Bhattacharyya, J. S. Nath, and S. Raman, "Variable Sparsity Kernel Learning," *Journal of Machine Learning Research*, vol. 12, pp. 565–592, 2011.

[45] F. Bach, "Exploring Large Feature Spaces with Hierarchical Multiple Kernel Learning," in *NIPS 21*, 2009.

[46] J. Yang, Y. Li, Y. Tian, L.-Y. Duan, and W. Gao, "Per-Sample Multiple Kernel Approach for Visual Concept Learning," *EURASIP Journal on Image and Video Processing*, vol. 2010, pp. 1–13, January 2010.

[47] M. Gnen and E. Alpaydin, "Localized Multiple Kernel Learning," in *ICML*, 2008.

[48] S. Ji, L. Sun, R. Jin, and J. Ye, "Multi-label Multiple Kernel Learning," in *NIPS 21*, 2009.

[49] M. Varma and D. Ray, "Learning the Discriminative Power-invariance Trade-Off," in *ICCV*, 2007.

[50] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large Scale Multiple Kernel Learning," *Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[51] H. Liu and L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 491–502, 2005.

[52] F. R. Bach, "Consistency of the Group Lasso and Multiple Kernel Learning," *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, 2008.

[53] T. Hertz, "Learning Distance Functions: Algorithms and Applications," Ph.D. dissertation, The Hebrew University of Jerusalem, 2006.

[54] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, "On Kernel-Target Alignment," in *NIPS 14*, 2002, pp. 367–373.

[55] O. Chapelle, J. Weston, and B. Schlkopf, "Cluster Kernels for Semi-supervised Learning," in *NIPS 15*, 2003, pp. 585–592.

[56] R. I. Kondor and J. Lafferty, "Diffusion Kernels on Graphs and Other Discrete Structures," in *ICML*, 2002, pp. 315–322.

[57] J. Zhuang, I. W. Tsang, and S. C. H. Hoi, "SimpleNPKL: Simple Non-Parametric Kernel Learning," in *ICML*, 2009.

[58] B. Kulis, M. Sustik, and I. Dhillon, "Learning Low-rank Kernel Matrices," in *ICML*, 2006, pp. 505–512.

[59] S. C. H. Hoi and R. Jin, "Active Kernel Learning," in *ICML*, 2008, pp. 400–407.

[60] S. Dickinson, "The Evolution of Object Categorization and the Challenge of Image Abstraction," in *S. Dickinson, A. Leonardis, B. Schiele, and M. Tarr, (eds), Object Categorization: Computer and Human Vision Perspectives*. Cambridge University Press, 2009, pp. 1–37.

[61] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman, "Dataset Issues in Object Recognition," in *Toward Category-Level Object Recognition*, ser. Lecture Notes in Computer Science, 2006, pp. 29–48.

[62] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, 1988.

[63] K. Mikolajczyk and C. Schmid, "Indexing Based on Scale Invariant Interest Points," in *ICCV*, 2001, pp. 525–531.

[64] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[65] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution Grayscale and Rotation Invariant Texture Classification with Local Binary Patterns," *IEEE PAMI*, vol. 24(7), pp. 971–987, 2002.

[66] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 886–893 vol. 1.

[67] F. Perronnin, J. Snchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*, 2010.

[68] C. Schmid, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006, pp. 2169–2178.

[69] A. Oliva and A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," *International Journal of Computer Vision*, vol. 42(3), pp. 145–175, 2001.

[70] N. Pinto, D. D. Cox, and J. J. Dicarlo, "Why is Real-World Visual Object Recognition Hard," *PLoS Computational Biology*, vol. 4(1), 2008.

[71] A. Berg and J. Malik, "Geometric Blur for Template Matching," in *CVPR*, 2001, pp. 607–614.

[72] V. Sindhwani and A. C. Lozano, "Non-parametric Group Orthogonal Matching Pursuit for Sparse Learning with Multiple Kernels," in *NIPS 24*, 2011, pp. 414–431.

[73] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 1998.

[74] L. Tang, J. Chen, and J. Ye, "On Multiple Kernel Learning with Multiple Labels," in *IJCAI*, 2009.

[75] A. Zien and S. Cheng, "Multiclass Multiple Kernel Learning," in *ICML*, 2007.

[76] F. R. Bach, "Consistency of the Group Lasso and Multiple Kernel Learning," *Journal of Machine Learning Research*, vol. 9, pp. 1179–1225, 2008.

[77] [Online]. Available: http://www.mosek.com

[78] J. C. Platt, *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[79] R. Jin, S. C. H. Hoi, and T. Yang, "Online Multiple Kernel Learning: Algorithms and Mistake Bounds," in *International Conference on Algorithmic Learning Theory*, 2010, pp. 390–404.

[80] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[81] Y. Freund and R. E. Schapire, "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[82] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[83] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE PAMI*, vol. 24, pp. 509–522, 2002.

[84] O. Tuzel, F. Porikli, and P. Meer, "Human Detection via Classification on Riemannian Manifolds," in *CVPR*, 2007, pp. 1–8.

[85] E. Shechtman and M. Irani, "Matching Local Self-similarities Across Images and Videos," in *CVPR*, 2007, pp. 607–614.

[86] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, "TagProp: Discriminative Metric Learning in Nearest Neighbor Models for Image Auto-annotation," in *ICCV*, sep 2009, pp. 309–316.

[87] J. van de Weijer and C. Schmid, "Coloring Local Feature Extraction," in *ECCV*, 2006, pp. 334–348.

[88] F. Perronnin, J. Sanchez, and Y. Liu, "Large-Scale Image Categorization with Explicit Data Embedding," in *CVPR*, 2010, pp. 2297–2304.

[89] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[90] M. Grant and S. Boyd., "CVX: Matlab Software for Disciplined Convex Programming, version 1.21," http://cvxr.com/cvx, april 2011.

[91] F. Bach, R. Thibaux, and M. I. Jordan, "Computing Regularization Paths for Learning Multiple Kernels," in *NIPS 17*, 2005.

[92] F. Li, J. Carreira, and C. Sminchisescu, "Object Recognition as Ranking Holistic Figure-ground Hypotheses," in *CVPR*, 2010, pp. 1712 – 1719.

[93] G. L. Oliveira, E. R. Nascimento, A. W. Vieira, and M. F. M. Campos, "Sparse Spatial Coding: A Novel Approach for Efficient and Accurate Object Recognition," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2592–2598.

[94] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing Object Detection and Classification," in *CVPR*, 2011, pp. 1585 – 1592.

[95] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *International Conference on Computer Vision*, 2009, pp. 237–244.

[96] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *British Machine Vision Conference*, 2011.

[97] S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc, "The SHOGUN machine learning toolbox," *Journal of Machine Learning Research*, vol. 11, pp. 1799–1802, Jun. 2010.

Serhat Selcuk Bucak received the BS and MS degrees from the Electronics & Communication Engineering Department of Istanbul Technical University, in 2006 and 2008, respectively. Currently, he is working toward the PhD degree in the Department of Computer Science and Engineering, Michigan State University. His research interests include pattern recognition, computer vision and machine learning.

Rong Jin focuses his research on statistical machine learning and its application to information retrieval. He has worked on a variety of machine learning algorithms and their application to information retrieval, including retrieval models, collaborative filtering, cross lingual information retrieval, document clustering, and video/image retrieval. He has published over 180 conference and journal articles on related topics. Dr. Jin holds a Ph.D. in Computer Science from Carnegie Mellon University. He received the NSF Career Award in 2006.

Anil K. Jain is a university distinguished professor in the Department of Computer Science and Engineering at Michigan State University, East Lansing. His research interests include pattern recognition and biometric authentication. He served as the editor-in-chief of the IEEE Transactions on Pattern Analysis and Machine Intelligence (1991-1994). He is the coauthor of a number of books, including Handbook of Fingerprint Recognition (2009), Handbook of Biometrics (2007), Handbook of Multibiometrics (2006), Handbook of Face Recognition (2005), BIOMETRICS: Personal Identification in Networked Society (1999), and Algorithms for Clustering Data (1988). He served as a member of the Defense Science Board and The National Academies committees on Whither Biometrics and Improvised Explosive Devices. He received the 1996 IEEE Transactions on Neural Networks Outstanding Paper Award and the Pattern Recognition Society best paper awards in 1987, 1991, and 2005. He has received Fulbright, Guggenheim, Alexander von Humboldt, IEEE Computer Society Technical Achievement, IEEE Wallace McDowell, ICDM Research Contributions, and IAPR King-Sun Fu awards. He is a fellow of the AAAS, ACM, IAPR, SPIE, and IEEE.