On Feature Combination for Multiclass Object Classification

Peter Gehler and Sebastian Nowozin
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
{peter.gehler,sebastian.nowozin}@tuebingen.mpg.de

Abstract

A key ingredient in the design of visual object classification systems is the identification of relevant class specific aspects while being robust to intra-class variations. While this is a necessity in order to generalize beyond a given set of training images, it is also a very difficult problem due to the high variability of visual appearance within each class. In the last years substantial performance gains on challenging benchmark datasets have been reported in the literature. This progress can be attributed to two developments: the design of highly discriminative and robust image features and the combination of multiple complementary features based on different aspects such as shape, color or texture. In this paper we study several models that aim at learning *the correct weighting of different features from training data. These include multiple kernel learning as well as simple baseline methods. Furthermore we derive ensemble methods inspired by Boosting which are easily extendable to several multiclass setting. All methods are thoroughly evaluated on object classification datasets using a multitude of feature descriptors. The key results are that even very simple baseline methods, that are orders of magnitude faster than learning techniques are highly competitive with multiple kernel learning. Furthermore the Boosting type methods are found to produce consistently better results in all experiments. We provide insight of when combination methods can be expected to work and how the benefit of complementary features can be exploited most efficiently.*

1. Introduction

In this paper we address the problem of object category classification by combining multiple diverse feature types. For a given test image the learned classifier has to decide which class the image belongs to. This problem is challenging because the instances belonging to the same class usually have high intraclass variability.

To overcome the problem of variability, one strategy is to design feature descriptors which are highly invariant to the variations present within the classes. Invariance is an improvement, but it is clear that none of the feature descriptors will have the same discriminative power for all classes. For example, features based on color information might perform well when classifying leopards or zebras, whereas a classifier for cars should be invariant to the actual color of the car. Therefore it is widely accepted that, instead of using a single feature type for all classes it is better to adaptively *combine* a set of diverse and complementary features – such as features based on color, shape and texture information – in order to discriminate each class best from all other classes.

Finding these feature combinations is a recent trend in class-level object recognition and image classification. One popular method in computer vision is Multiple Kernel Learning (MKL), originally proposed in [9]. In the application of MKL to object classification, the approach can be seen to linearly combine similarity functions between images such that the combined similarity function yields improved classification performance [8, 11, 20].

In Section 2 we give a general overview of the problem addressed in this paper. The Sections 3-5 describes several combination approaches. Experiments are presented in Section 6 and 7. We conclude with a discussion in Section 8.[1]

2. Feature Combination Methods

We begin with a formal definition of the problem we address in this paper.

Definition 1 (Feature Combination Problem) *Given a training set $\{(x_i, y_i)\}_{i=1,\ldots,N}$ of N instances consisting of an image $x_i \in \mathcal{X}$ and a class label $y_i \in \{1,\ldots,\mathcal{C}\}$, and given a set of F image features $f_m : \mathcal{X} \to \mathbb{R}^{d_m}$, $m = 1,\ldots,F$ where d_m denotes the dimensionality of the m'th feature, the problem of learning a classification function $y : \mathcal{X} \to \{1,\ldots,\mathcal{C}\}$ from the features and training set is called* feature combination problem.

[1] The code and scripts used to produce the results in this paper are available online at http://www.vision.ee.ethz.ch/~pgehler/

A typical example of such a feature f_m would be a bag-of-visual-words histogram of the image. Then, the corresponding dimensionality d_m would be the codebook size used for the vector quantization step. In the following, we will use the name *feature combination method* for all methods which address the feature combination problem.

Kernel methods. The object classification problem is a special case of multiclass classification. In computer vision the problem of learning a multiclass classifier from training data is often addressed by means of *kernel methods*. Kernel methods make use of kernel functions defining a measure of similarity between pairs of instances. In the context of feature combination it is useful to associate a kernel to each image feature as follows. For a kernel function k between real vectors we define the short-hand notation

$$k_m(x, x') = k(f_m(x), f_m(x')),$$

such that the image kernel $k_m : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ only considers similarity with respect to image feature f_m. If the image feature is specific to a certain aspect, say, it only considers texture information, then the kernel measures similarity only with regard to this aspect. The subscript m of the kernel can then be understood as indexing into the set of features.

In the following, for notational convenience, we will denote the kernel response of the m'th feature for a given sample $x \in \mathcal{X}$ to all training samples x_i, $i = 1, \ldots, N$ as $K_m(x) \in \mathbb{R}^N$ with

$$K_m(x) = [k_m(x, x_1), k_m(x, x_2), \ldots, k_m(x, x_N)]^T.$$

In case x is the i'th training sample, *i.e.* $x = x_i$, then $K_m(x)$ is simply the i'th column of the m'th kernel matrix.

Feature selection as kernel selection In this paper we study a class of kernel classifiers that aim to combine several kernels into a single model. Since we associate image features with kernel functions, kernel combination/selection translates naturally into feature combination/selection.

A conceptually simple approach is the use of Cross Validation (CV) to select a single kernel from the set $\{k_1, \ldots, k_F\}$. Every feature combination method should be able to outperform this baseline method or at least match its performance if a single feature is sufficient for good classification.

In the following we will present several methods in a unified setting along with their training procedures. An overview of the different methods in their multiclass variant can also be found in the Table 1.

3. Methods: Baselines

We include two simple baseline methods, both of which combine kernels in a pre-defined deterministic way and subsequently use the resulting kernel for SVM training.

3.1. Averaging Kernels

Arguably the simplest method to combine several kernels is to average them. We define the kernel function $k^*(x, x') = \frac{1}{F} \sum_{m=1}^{F} k_m(x, x')$, which is subsequently used in a support vector machine (SVM).

Training The only free parameters are the SVM parameters. We use CV to estimate the best regularization constant. A multiclass variant is build using a one-versus-all scheme.

3.2. Product Kernels

The next baseline method we consider is to combine several kernels by multiplication. In this case we use $k^*(x, x') = (\prod_{m=1}^{F} k_m(x, x'))^{1/F}$ as the single kernel in a SVM.

Training Same as for averaging.

4. Methods: Multiple Kernel Learning

Another approach to perform kernel selection is to learn a kernel combination during the training phase of the algorithm. One prominent instance of this class is MKL. Its objective is to optimize *jointly* over a linear combination of kernels $k^*(x, x') = \sum_{m=1}^{F} \beta_m k_m(x, x')$ and the parameters $\alpha \in \mathbb{R}^N$ and $b \in \mathbb{R}$ of an SVM.

MKL was originally introduced in [1]. For *efficiency* and in order to obtain sparse, *interpretable* coefficients, it restricts $\beta_m \geq 0$ and imposes the constraint $\sum_{m=1}^{F} \beta_m = 1$. Since the scope of this paper is to access the applicability of MKL to feature combination rather than its optimization part we opted to present the MKL formulations in a way allowing for easier comparison with the other methods. We write its objective function as

$$\begin{aligned} \min_{\alpha,\beta,b} \quad & \frac{1}{2} \sum_{m=1}^{F} \beta_m \alpha^T K_m \alpha \\ & + C \sum_{i=1}^{N} L(y_i, b + \sum_{m=1}^{F} \beta_m K_m(x)^T \alpha) \\ \text{sb.t.} \quad & \sum_{m=1}^{F} \beta_m = 1, \qquad \beta_m \geq 0, \quad m = 1, \ldots, F, \end{aligned} \tag{1}$$

where $L(y, t) = \max(0, 1 - yt)$ denotes the Hinge loss. We compare two different algorithms solving this problem for their runtime performance, namely SILP [18][2] and SimpleMKL [17][3].

The final binary decision function of MKL is of the following form

$$F_{\text{MKL}}(x) = \text{sign}\left(\sum_{m=1}^{F} \beta_m (K_m(x)^T \alpha + b) \right). \tag{2}$$

[2]Available online:`www.shogun-toolbox.org/`

[3]Available online:`mloss.org/software/view/174/`

Name	Test-time function	Coefficients	Training	Parameters	References
Averaging	$y(x) = \underset{c=1,\dots,\mathcal{C}}{\operatorname{argmax}} [\left(\frac{1}{F}\sum_{m=1}^{F} K_m(x)\right)^T \alpha_c + b_c]$	$\alpha \in \mathbb{R}^{\mathcal{C}\times N}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha, b)_c$, ind.	C_c	
Product	$y(x) = \underset{c=1,\dots,\mathcal{C}}{\operatorname{argmax}} [\left(\left(\prod_{m=1}^{F} K_m(x)\right)^{1/F}\right)^T \alpha_c + b_c]$	$\alpha \in \mathbb{R}^{\mathcal{C}\times N}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha, b)_c$, ind.	C_c	
MKL	$y(x) = \underset{c=1,\dots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^{F} \beta_m^c \left(K_m(x)^T \alpha_c + b_c\right)$	$\beta \in \mathbb{R}^{\mathcal{C}\times F}$ $\alpha \in \mathbb{R}^{\mathcal{C}\times N}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha_c, b_c, \beta^c)_c$ ind.	C_c	[20, 18, 1]
CG-Boost	$y(x) = \underset{c=1,\dots,\mathcal{C}}{\operatorname{argmax}} [\sum_{m=1}^{F} K_m(x)^T \alpha_{c,m} + b_c]$	$\alpha \in \mathbb{R}^{\mathcal{C}\times F\times N}$ $b \in \mathbb{R}^{\mathcal{C}}$	$(\alpha, b)_c$, ind.	C_c	[2]
LP-β	$y(x) = \underset{c=1,\dots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^{F} \beta_m \left(K_m(x)^T \alpha_{c,m} + b_{c,m}\right)$	$\beta \in \mathbb{R}^{F}$ $\alpha \in \mathbb{R}^{\mathcal{C}\times F\times N}$ $b \in \mathbb{R}^{\mathcal{C}\times F}$	1. $(\alpha, b)_c$, ind 2. β, jointly	1. C_m 2. $\nu \in (0, 1)$	[4]
LP-B	$y(x) = \underset{c=1,\dots,\mathcal{C}}{\operatorname{argmax}} \sum_{m=1}^{F} B_m^c \left(K_m(x)^T \alpha_{c,m} + b_{c,m}\right)$	$B \in \mathbb{R}^{F\times\mathcal{C}}$ $\alpha \in \mathbb{R}^{\mathcal{C}\times F\times N}$ $b \in \mathbb{R}^{\mathcal{C}\times F}$	1. $(\alpha, b)_c$, ind 2. B, jointly	1. C_m, 2. $\nu \in (0, 1)$	[4]

Table 1. Comparison of multiclass learning approaches to the feature combination problem in image and object classification. In the column "Training" it is also noted which parameters are trained independently (ind.) over the classes c and which are trained jointly.

A slightly different MKL variant was proposed in [20] where the norm-1 constraint on β is replaced with an extra term in the objective function. Although this formulation is different to (1) we empirically found that in case of strong regularization both yield exactly the same solution. Since high values of C turn out to work best for the experiments in this paper (consistent with the results of [20] where C=1000 is fixed) both variants can be regarded as being equal.

Training The only free parameter in the MKL approaches is the regularization constant C, which is chosen using CV. A multiclass decision is resolved with a one-versus-rest scheme, see Table 1 for the final decision function. All one-versus-rest classifiers can be trained in parallel.

Multiclass MKL For strongly unbalanced datasets a MKL classifier trained as a multiclass classifier might be preferable over the one-versus-rest setup. The authors of [22] derive such a MC-MKL formulation in which all parameters for all classes are trained jointly. Due to performance issues this approach renders infeasible for the experiments presented here.[4]

5. Methods: Boosting Approaches

As last class of feature combination methods we look at boosting approaches and in particular propose two methods which are inspired by the MKL decision function. All methods in this section are, as MKL, based on mixture of kernels.

5.1. LPBoost

With the mixing coefficients β_m summing to one, the MKL decision function is a convex linear combination of the real valued output of F SVMs $f_m(x) = K_m(x)^T\alpha + b$. Furthermore we observe that all of the SVMs included in the sum share the same parameter set $\{\alpha, b\}$. Having noted this, MKL can be understood as a restricted version of the following more general form

$$F(x) = \operatorname{sign} \sum_{m=1}^{F} \beta_m f_m(x), \tag{3}$$

where $f_m(x)$ are some real valued functions, not necessarily support vector machines and not necessarily trained jointly. In Boosting terminology the f_m are also known as *weak learners*.

This observation leads naturally to the following model. We use the kernels k_m to train F separate SVMs f_m, resulting in different parameter sets $\{\alpha_m, b_m\}$. Subsequently we optimize over β in a second step. Each individual function f_m is not restricted to share the parameters but can be trained to yield maximal generalization. The details of this two-step learning procedure are given in Section 5.2.

This procedure is searching over the same hypothesis space as MKL, allowing for multiple α does not enhance the space of possible decision functions. The resulting decision function of this approach is a convex combination of several hyperplanes and thus itself a hyperplane. If it were the optimal one for the MKL problem it could, due to the representer theorem, be represented as a combination of the kernel evaluations using only N training points. The training procedure proposed here with separate training of the participating SVMs is a different regularization of the same hypothesis space.

Problem formulation We propose to learn all parameters of the model in two separate steps. In the first step the func-

[4] Personal correspondence with the authors.

tions f_m are trained individually. Subsequently we optimize over β using the following linear program, which is equivalent to ν-LPBoost [4]

$$\begin{aligned} \min_{\beta,\xi} \quad & -\rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \\ \text{sb.t.} \quad & y_i \sum_{m=1}^{F}\beta_m f_m(x_i) + \xi_i \geq \rho, \quad i = 1,\ldots,N \\ & \sum_{m=1}^{F}\beta_m = 1, \quad \beta_m \geq 0, \quad m = 1,\ldots,F, \end{aligned} \tag{4}$$

with ξ being slack variables. The equivalence to LPBoost can be seen by considering the hypothesis space to be the finite set of functions $\{f_1, f_2, \ldots, f_F\}$. The problem can easily be solved using standard linear programming solvers.[5] There is only one hyperparameter ν in the problem which trades the smoothness of the resulting function with the hinge loss on the points, analogously to the SVM regularization parameter C.

5.2. Multiclass LPboost variants

It is straightforward to derive a multiclass version of Problem (4). In the multiclass case with $\mathcal{C}$ classes, the functions f_m are no longer real-valued but map into a $\mathcal{C}$ dimensional space $f_m(x) \to \mathbb{R}^{\mathcal{C}}$. The c'th output of f_m will be denoted by $f_{m,c}(x)$.

We consider two possible variations of learning feature weights. The first, termed *LP-β* uses a single vector β for all classes. This β defines a combination that works well for all classes *jointly*. Alternatively each class can have its own weight vector over the features, in which case there is a weight matrix $B \in \mathbb{R}^{F\times\mathcal{C}}$, we name this method *LP-B*.

LP-β. The decision rule of LP-β can be found in Table 1. Again we train the parameters (α_m, b_m) of f_m for all classes and features in a first step. The mixing coefficients β are learned by the following multiclass extension of LPBoost.

$$\begin{aligned} \min_{\beta,\xi,\rho} \quad & -\rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \\ \text{sb.t.} \quad & \sum_{m=1}^{F}\beta_m f_{m,y_i}(x_i) - \operatorname*{argmax}_{y_j\neq y_i}\sum_{m=1}^{F}\beta_m f_{m,y_j}(x_i) \\ & \ldots + \xi_i \geq \rho \quad i = 1,\ldots,N \\ & \sum_{m=1}^{F}\beta_m = 1, \quad \beta_m \geq 0,\ m = 1,\ldots,F \end{aligned} \tag{5}$$

[5] Due to the special coupled structure in the dense constraint matrix, we found interior-point based solvers to be consistently faster than simplex based method. We use the MOSEK interior-point solver, see `www.mosek.com`.

Since we are only optimizing over $\mathcal{C}$ parameters, learning them in such a true multiclass formulation is feasible. Furthermore we do not need worry about normalization of the kernels which is an inherent problem of MKL. Another benefit of this method is that β is sparse on the level of the features. This means features for which $\beta_m = 0$ need not to be computed for the final decision function. Although the MKL solution is sparse for every class separately, it is not sparse *jointly* and the one-versus-rest MKL setup and almost always every feature is selected at least once.

LP-B In this second variant each class is assigned its own weight vector resulting in a $F \times \mathcal{C}$ weighting matrix B. The decision rule is shown in Table 1. The corresponding learning problem extends the Hinge loss in (5) to the multiclass Hinge loss originally proposed by Weston and Watkins [21] for Support Vector Machines.

$$\begin{aligned} \min_{B,\xi,\rho} \quad & -\rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \\ \text{sb.t.} \quad & \sum_{m=1}^{F}B_m^{y_i} f_{m,y_i}(x_i) - \sum_{m=1}^{F}B_m^{y_j} f_{m,y_j}(x_i) \\ & \ldots + \xi_i \geq \rho \quad i = 1,\ldots,N,\ y_j \neq y_i, \\ & \sum_{m=1}^{F}B_m^c = 1,\ m = 1,\ldots,F,\ B_m^c \geq 0\ \forall m, c \end{aligned}$$

Note that this is still a linear programming problem, but more expensive to solve than LP-β due to the increased number of parameters.

Training The training procedure for LP-β and LP-B is analogous. Ideally we have enough data to adjust f_m and β on independent sets. Since this is usually not the case, we use the following two stage scheme to avoid biased estimates. First we perform model selection using 5 fold CV to select the best hyperparameters for each f_m individually (in our case where f_m are SVMs we need to select C). At this point the only parameter left is ν. Since there is no independent training data left to set this parameter we compute for each f_m the CV outputs using its best hyperparmeter identified before. This results in a prediction for each training point using a classifier which was *not* trained using that point (but on other 80% of the training data). The CV outputs of all SVMs f_m are used as training data for LP-β. We perform CV to select the best parameter ν and subsequently train the final combination β. The main concern using this scheme, is that the input to the LP-β training is not from the classifier f_m later used in the combination. However it is reasonable to assume that the learners used to produce the training data for LP-β are not too different. The experiments validate this assumption as we do not observe overfitting for the LP-β model. The LP-B results tend to be

Figure 1. Ten example images from the Flowers dataset. Images in the same column are from the same class.

Single features			Combination methods		
Method	Accuracy	Time	Method	Accuracy	Time
Colour	60.9 ± 2.1	3	product	85.5 ± 1.2	2
Shape	70.2 ± 1.3	4	averaging	84.9 ± 1.9	10
Texture	63.7 ± 2.7	3	CG-Boost	84.8 ± 2.2	1225
HOG	58.5 ± 4.5	4	MKL (SILP)	85.2 ± 1.5	97
HSV	61.3 ± 0.7	3	MKL (Simple)	85.2 ± 1.5	152
siftint	70.6 ± 1.6	4	LP-β	85.5 ± 3.0	80
siftbdy	59.4 ± 3.3	5	LP-B	85.4 ± 2.4	98

Table 2. Mean accuracy for all methods on the Oxford Flowers dataset using the predefined splits [14]. Also plotted is the total time for model selection, training and testing in seconds.

worse compared to LP-β, so fitting its $\mathcal{C} \times F$ instead of F parameters demands for more training data.

5.3. Column generation Boosting for mixtures of kernels

A different variant of Boosting technique which is based on the same observation from 5.1 was proposed in [4]. Instead of maintaining the separation between the SVM parameters (α, b) and mixing coefficients β the authors propose to solve

$$\min_{\alpha_m} \quad \frac{1}{2}\sum_{m=1}^{F} \alpha_m^T \alpha_m + C \sum_{i=1}^{N} L(y_i, b + \sum_{m=1}^{F} K_m(x_i)^T \alpha_m).$$

This formulation can be understood as training a SVM with a linear kernel with the kernel evaluations at the training points as features.[6] This method is referred to as CG-Boost in the experiments.

Training Since the formulation reduces to a linear SVM the only free parameter is again the regularization parameter C. It is selected using CV. We experimented with different loss functions L and found that logistic regression yields best results while assuring good convergence of the algorithm.

6. Experiments: Oxford Flowers

In this section we present results on the Oxford flowers dataset [13]. This dataset consists of flower images depicting 17 different types with 80 images per category. Example images are shown in Figure 1.

[6]Implemented using liblinear-1.33, a standard solver for linear SVM.

The dataset comes with three predefined splits into test (17×20 images), train (17×40 images) and validation set (17×20 images). Furthermore the authors of [14] provide seven precomputed distance matrices online on their website[7] and those matrices are used for the experiments presented here. Each matrix is computed using a different feature type, namely clustered HSV values, SIFT features on the foreground region, SIFT features on the foreground boundary and three matrices derived from colour, shape and texture vocabularies. For brevity we omit details of the features and refer to [13, 14].

We first compare the overall performance of all models presented in this paper. To this end we use the predefined splits for training and model selection. The regularization parameter is selected from the range $C \in \{0.01, 0.1, 1, 10, 100, 1000\}$. For LP-$\beta$ and LP-B the regularization parameter $\nu \in \{0.05, 0.1, \ldots, 0.95\}$ of the second stage is also selected on the validation set, using the procedure described in 5.2. Kernel matrices are computed as $\exp(-\gamma^{-1} d(x, x'))$ with d being the distance and γ being fixed to the mean of the pairwise distances.

The results are shown in Table 2, with results using a SVM with single kernel only are shown in the left, and combination methods in the right column. From the results we can draw several conclusions. All feature combination methods dramatically improve the classification performance. This is consistent with [14] whose results we herewith restated. Having said this, we note that the single baselines which are almost always left out in comparisons (*e.g.* [14, 20]) yield equally good results but are magnitudes faster than any other combination method.

These results also shed a new light on the interpretability of the MKL solution which is mentioned by several authors. The mixing coefficients of the MKL solution are usually interpreted as the influence of the features on a particular class. In the multiclass setting this reasoning is misleading. One could equally plausible argue, that due to the good result of the averaging baseline all features are equally important for the multiclass decision. LP-β selects only three out of the seven features while all other methods select every feature at least once.

As a second experiment we will highlight the benefit of using a learning approach for feature combination over the baseline methods. Additional to the seven discriminative features we add non-discriminative features by generating random vectors from a three dimensional isotropic Gaussian distribution. A Gaussian kernel is computed on these noise features and included into the set of kernels. Now the same experiment as before is repeated. In Figure 3 we plot the performance of the models against the number of added noise kernels. The baselines incorporate all features and

[7]`www.robots.ox.ac.uk/~vgg/research/flowers/index.html`

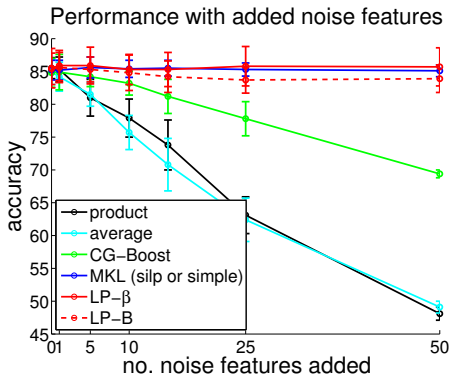

Table 3. Accuracy of the different methods with several noise features included. Learning based combination methods are robust to the inclusion of noise features, while baseline techniques are not.

subsequently their performance drops severely. Among the learning methods MKL and LP-β turn out to be very robust to uninformative features, while the CG-Boosting approach slowly decays in performance.

This experiment highlights a feature of MKL and the boosting technique, namely that it is possible to select kernels out of a large class of potentially un-informative ones, *e.g.* wrong kernel parameters. This may not be critical in terms of performance if each participating feature is individually designed to be discriminative.

7. Experiments: Caltech datasets

For the second set of experiments we use the well known Caltech datasets [5, 7] which are prominent benchmark datasets for object classification. We follow the experimental setup proposed by the designers of the datasets. Performance is measured as the mean prediction rate per class, thus balancing the influence of categories with a large number of test examples. We report results using all 102 classes of the Caltech101 dataset averaged over three splits and for 256 classes of the Caltech256 dataset, excluding its clutter category for a single split. The number of training images is varied using $5, 10, 15, 20, 25, 30$ images per category for training and up to 50 images per category for testing. For the Caltech256 dataset 25 test images per category are used.

All participating kernels are of the form $k(x, x') = \exp(-\gamma^{-1} d(x, x'))$ with γ selected as before. For the shape, appearance and LBP descriptors the χ^2 distance is used, whereas all other kernels are computed using a Gaussian kernel. The regularization parameter C is chosen from the set $\{0.1, 1, 10, 50, 100, 500, 1000\}$ and ν in the range of 0.05 to 0.95 in steps of 0.05. Values around 0.8 are typical. For MKL we fix $C = 1000$ which yields best results.

7.1. Image Descriptors

In the following we give an overview of the features which were used for the experiments. We compute all but the V1S+ features in a spatial pyramid as proposed in [10]. A pyramid representation consists of several levels which themselves consist of several *cells*. The first level 0 of the pyramid is the image itself and in each subsequent level each cell is split into four non-overlapping windows. This process is repeated up to level L and for each level the features of all its are concatenated to build the final descriptor. These vectors are used to generate kernels for each level which we refer to as pyramid kernels.

Analog to the spatial pyramid we compute a kernel in the way proposed by [6]. A subwindow is drawn randomly and a histogram of SIFT features which fall into this subwindow is computed. All such histograms define a new image feature for which a kernel is computed. This process is repeated 100 times and the resulting 100 kernels are finally averaged yielding a kernel we refer to as subwindow-kernel.

In the following we briefly describe the image features used for the experiments and refer to the corresponding publications for more details.

PHOG Shape Descriptor. Shape is modeled using the PHOG descriptor proposed in [3]. The descriptor is a histogram of oriented (Shp_{360}) or unoriented (Shp_{180}) gradients computed on the output of a Canny edge detector. The oriented histogram Shp_{360} contains 40 bins, the unoriented Shp_{180} 20 bins yielding a total of 2×4 kernels (L=3).

Appearance Descriptor. Appearance information is modeled using SIFT descriptors [12] which are computed on a regular grid on the image with a spacing of 10 pixels and for the four different radii $r = 4, 8, 12, 16$. The descriptors are subsequently quantized into a vocabulary of visual words that is generated by k-means clustering. We use four variants: two codebook sizes (300 and 1000 elements) and grey image descriptors (128 dims) as well as HSV-SIFT (3*128=384dims)) with a total of 4×4 kernels (L=3)

Region Covariance. We use the covariances of simple per-pixel features described in [19] (tangent-space projected). A pyramid representation yields 3 kernels (L=2).

Local Binary Patterns. Ojala et al. [15] argue to use *locally binary pattern* (LBP) features, retaining the classification performance of textons while being much faster and simpler to extract. We use histograms of uniform rotation-invariant $LBP_{8,1}$-features and create 3 kernels (L=2).

V1S+ In [16] a population of locally normalized, thresholded Gabor functions spanning a range of orientations and spatial frequencies are derived and advocated as particular simple features. This generates one kernel (L=0).

7.2. Results

We distinguish two different settings: combining kernels based on the same feature type, *i.e.* different levels of a

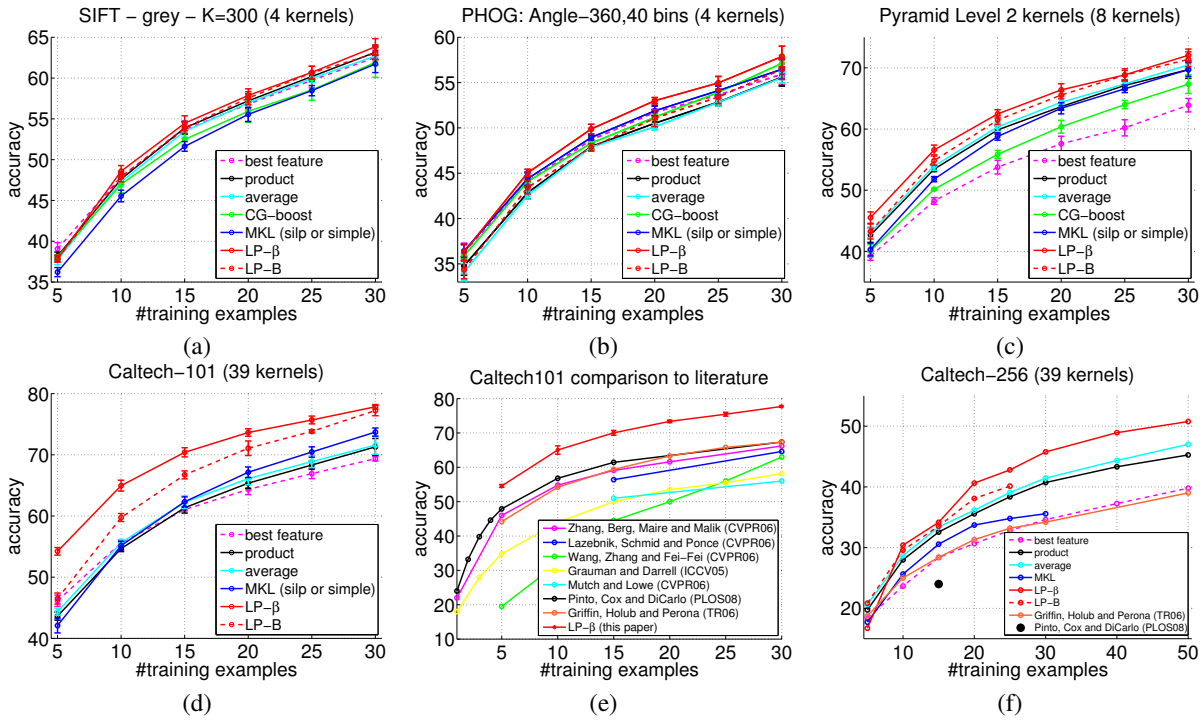

Figure 2. Performance of combination methods on the Caltech datasets. (a)+(b) results combining four kernels of a spatial pyramid using the same image feature type (SIFT or PHOG). (c) combining eight kernels of different image features. (d) comparison of methods using a total of 39 kernels, (e)+(f) same as (d) but with a comparison to other published results. The exact numbers are stated in the supplementary material. Plot best viewed in color.

pyramid and combining diverse features *e.g*. different types of image features.

In Figure 2 results are shown for combining the four kernels of the spatial pyramid using SIFT (a) or PHOG (b) features. The dotted magenta line corresponds to the result of the best single kernel selected by CV. All models yield similar results but several observations can be made.

- Combining pyramid kernels of SIFT features MKL and CGBoost are outperformed by baseline methods and even yield worse results than using a single kernel alone (Fig.(a)). For combination of PHOG pyramid kernels the baseline results are worse than CGBoost or MKL.

- LP-β yields the best results for both combinations if trained with more than five examples.

Figure 2(c) shows the result of combining 8 kernels corresponding to different image features (Level 2 of every feature described previously) which are more diverse than combining levels of a pyramid.

- All feature combinations yield a significant improvement over the result using the single best kernel.

- CG-Boost and MKL are outperformed by the baseline methods.

In the next experiment we aim for maximal performance on the Caltech datasets using a total of 39 different kernels. In addition to the kernels from Section 7.1 we include the products of the pyramid levels for each feature resulting in 7 kernels. Furthermore we use two subwindow kernels with SIFT and HSV-SIFTs (codebooksize 1000). The best single feature for the Caltech101 dataset (selected using CV) are the V1S+ features. The result is shown in Figure2(d) and we observe the same behavior as before. The LPBoosting techniques yield best performance while the baselines and MKL are comparable. In Figure 2(e) we compare LP-β as the best method among the considered ones to several other results published in the literature.[8] Note that most of the results in (d) yield better performance than all competitors compared in (e). As mentioned already, LP-β identifies a sparse solution on the level of the features. For Caltech101 7 out of 39 features are selected (using 30 training examples) and 15 for Caltech256. The results for Caltech256 are shown in Figure2(f) with LP-β achieving a $> 10\%$ improvement over the best published result [7]. The results of LP-β for Caltech101 using 30 training images per category are 77.7 ± 0.3 and for Caltech256 45.8% (30 training images) and 50.8% (50 images per category).[9]

[8] We do not compare against [20, 3] since those results are erroneous. See *e.g*. supplementary material or authors websites.

[9] The numerical results of all experiments are included in the supplementary material.

7.3. Training time

Using 15 training examples per class which adds to a total of 1530 for Caltech101 (3840 for Caltech256) the required training time for an entire one-versus-rest SVM classifier using a single kernel is about 5s (50s for Caltech256). Estimating the 39 coefficients of β takes 60s (8.5m) and for B 935s (4.9h). A single run of LP-β requires about $6 * 39 * 5s + 60s \approx$ 21m ($6 * 39 * 50$s+8.5m$\approx$ 3.4h) including the computation of the CV output. The numbers are comparable to MKL training which takes about 23m (5h). Since we perform model selection for LP-β the actual training times are longer but can be controlled by the number of parameters one searches over. All implementations can most likely be optimized, we include these numbers to show that LP-β has a comparable runtime to MKL, whereas the baseline methods are orders of magnitudes faster. CGBoost was intractable for the Caltech256 dataset.

8. Conclusions

In this paper we studied several methods for feature combination. We interpreted the MKL decision function as a convex combination of SVMs and proposed formulations based on LPBoost. These are different to CG-Boost, in that they maintain two sets of parameters.

We found that the LP-β approach consistently outperforms all other considered methods. On both Caltech datasets we observe an $> 10\%$ improved performance over the best published result. We expect even better performance if we train with more image features or include other classification functions. Adding more learners comes with a reasonable additional cost since it only scales linearly in F, while any trained weak learners can be reused.

The two step training procedure arguably is less principled than a joint optimization. However in practice this seems not to be a problem and works well even in the case of few training examples. Due to the two training stages most of the training can be done in parallel with each piece being reasonable fast.

Most results turn out to be disadvantageous for MKL. The baseline methods yield competitive results and outperform MKL on several setups. This is due to the fact that the available kernels on their own are already discriminative. In the presence of un-informative kernels MKL as the LPBoosting techniques are able to identify a discriminative set of kernels and maintain good performance.

We conclude with the observation that the performance of MKL might have been overestimated in the past. The baseline methods "average" and "product" should be considered as its canonical competitors and included in any study using MKL. With LP-β we derived a method that yields better performance, is equally fast and leads to sparse multiclass object classification systems.

Acknowledgements

We thank Christoph Lampert for helpful discussions and the EU CLASS project IST 027978 for funding.

References

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

[2] J. Bi, T. Zhang, and K. P. Bennett. Column-generation boosting methods for mixture of kernels. In *KDD*, 2004.

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

[4] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *JMLR*, 2002.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRW*, page 178, 2004.

[6] P. V. Gehler and S. Nowozin. Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *CVPR*, 2009.

[7] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, Caltech, 2007.

[8] A. Kumar and C. Sminchisescu. Support kernel machines for object recognition. In *ICCV*, 2007.

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

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

[11] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh. Local ensemble kernel learning for object category recognition. In *ICCV*, 2007.

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

[13] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *ICCV*, pages 1447–1454, 2006.

[14] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[15] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE PAMI*, 24(7):971–987, 2002.

[16] N. Pinto, D. D. Cox, and J. J. Dicarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 2008.

[17] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *ICML*, 2007.

[18] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 7, 2006.

[19] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *CVPR*, 2007.

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

[21] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *ESANN*, 1999.

[22] A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *ICML*, pages 1191–1198. ACM Press, 06 2007.