DOI 10.1007/s11263-007-0084-6

Hybrid Generative-Discriminative Visual Categorization

Alex D. Holub · Max Welling · Pietro Perona

Received: 14 September 2005 / Accepted: 15 August 2007 / Published online: 18 October 2007

Abstract Learning models for detecting and classifying object categories is a challenging problem in machine vision. While discriminative approaches to learning and classification have, in principle, superior performance, generative approaches provide many useful features, one of which is the ability to naturally establish explicit correspondence between model components and scene features—this, in turn, allows for the handling of missing data and unsupervised learning in clutter. We explore a hybrid generative/discriminative approach, using 'Fisher Kernels' (Jaakola, T., et al. in Advances in neural information processing systems, Vol. 11, pp. 487–493, 1999), which retains most of the desirable properties of generative methods, while increasing the classification performance through a discriminative setting. Our experiments, conducted on a number of popular benchmarks, show strong performance improvements over the corresponding generative approach. In addition, we demonstrate how this hybrid learning paradigm can be extended to address several outstanding challenges within computer vision including how to combine multiple object models and learning with unlabeled data.

Keywords Machine learning · Object recognition · Discriminative learning · Support vector machines

A.D. Holub (✉) · P. Perona
Computation and Neural Systems, California Institute of Technology, MC 136-93, Pasadena, CA 91125, USA
e-mail: holub@vision.caltech.edu

M. Welling
Department of Computer Science, University of California Irvine, Irvine, CA 92697-3425, USA
e-mail: welling@ics.uci.edu

1 Introduction

Detecting and classifying objects and object categories in images is currently one of the most interesting, useful, and difficult challenges for machine vision. Much progress has been made during the past decade in formulating models that capture the visual and geometrical statistics of natural objects, in designing algorithms that can quickly match these models to images, and in developing learning techniques that can estimate these models from training images with limited supervision (Burl and Perona 1996; Ullman et al. 2002; Weber et al. 2000; Fergus et al. 2003; Lowe 2004; Dorko and Schmid 2005; Torralba et al. 2004; Leibe and Schiele 2004; Holub and Perona 2005). However, our best algorithms are not close to matching human abilities. Machine vision systems are at least two orders of magnitude worse than humans in several aspects including the number of categories that can be learned and recognized, the classification error rates, the classification speed, and the ease and flexibility with which new categories can be learned.

This work is motivated by the challenge of learning to recognize categories that look similar to one another. A number of methods have shown good performance on dissimilar categories (for example airplanes, automobiles, spotted cats, faces and motorcycles as in Fergus et al. 2003, Dorko and Schmid 2005). None of these methods has been shown to perform well on visual categories which look similar to one another such as bicycles and motorcycles or male and female faces. For example, while the 'constellation model' Fergus et al. (2003) has error rates of a few percent on dissimilar categories such as faces vs. airplanes and cars vs. cats, it has error rates around 30% if it is asked to recognize faces of different people (see the x axis of the plots in Fig. 1). Why does this discrepancy exist? As we shall see,

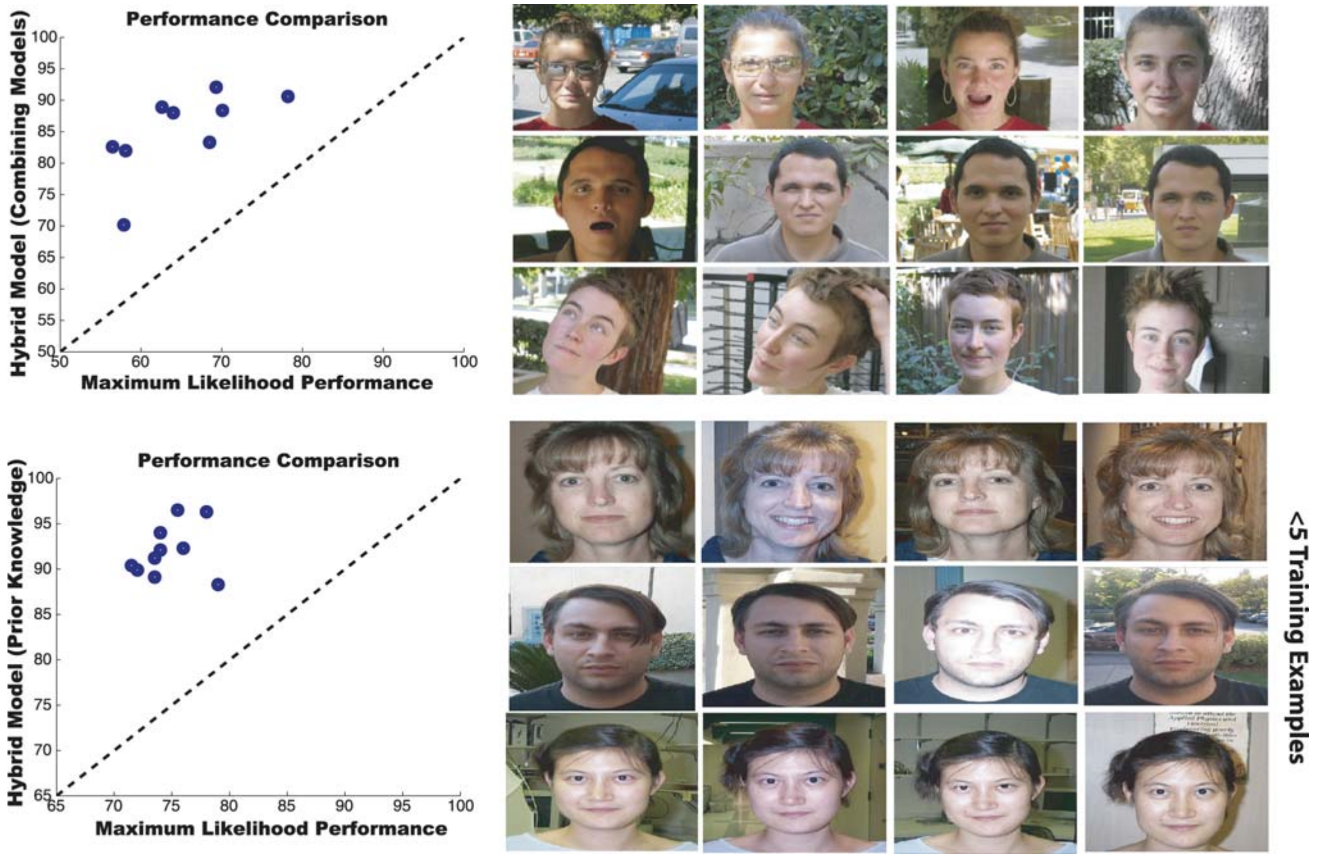

Fig. 1 A pure generative Maximum Likelihood (ML) approach will not work well when categories are similar in appearance (*right column* images of faces, each row shows a different person), especially when few training examples are available (scatterplots on the *left*, x axis). We apply discriminative techniques from Jaakkola et al. (1999) to transform generative approaches for visual recognition into discriminative classifiers which retain some of the desirable properties of generative models and yield better performance. (*Top*) ML in comparison to using combinations of hybrid models. See Sect. 6 below for details. Category labels for these faces, from top to bottom: P1, P2, and P3. These new face categories will be posted on the web. (*Bottom*) ML in comparison to hybrid models in a semi-supervised learning paradigm in which few examples (in this case three training examples) are present. See Sect. 5 below for details

one potential confound is the underlying generative learning algorithm. Note that code for this project is available at: www.vision.caltech.edu/IJCV_Holub_Hybrid_Code.zip.

Learning and classification methods fall into two broad categories (see Fig. 2). Let y be the label of the class and x the measured data associated with that class. A **generative** approach will estimate the joint probability density function $p(x, y)$ (or, equivalently, $p(x|y)$ and $p(y)$) and will classify using $p(y|x)$ which is obtained using Bayes' rule. Conversely, **discriminative** approaches will estimate $p(y|x)$ (or, alternatively, a classification function $y = f(x)$) directly from the data. It has been argued that the discriminative approach results in superior performance, i.e. why bother learning the details for models of different classes if we can directly learn a criteria for discriminating between the classes (Vapnik 1998)? Indeed, it was shown that the asymptotic (in the number of training examples) error of discriminative methods is lower than for generative ones when using simple learning models (Ng and Jordan 2002).

Yet, among machine vision researchers generative models remain popular (Ullman et al. 2002; Weber et al. 2000; Fergus et al. 2003; Dorko and Schmid 2005; Leibe and Schiele 2004; Schneiderman 2004). There are at least five good reasons why generative approaches are an attractive choice for visual recognition. First, generative models naturally incorporate information about occlusion and missing features. This is because generative methods allow one to establish explicit 'correspondence' between parts of the model and features in the image. For every such mapping, the parts in the model corresponding to the missing features can simply be marginalized out of the probabilistic model, leaving us with a lower dimensional model over the observed parts (Weber et al. 2000). Second, collecting training examples is expensive in vision and training sets come at a premium. Ng and Jordan (2002) demonstrated both analytically and experimentally that in a 2-class setting the generative approach often has better performance for small numbers of training examples, despite the asymptotic performance be-

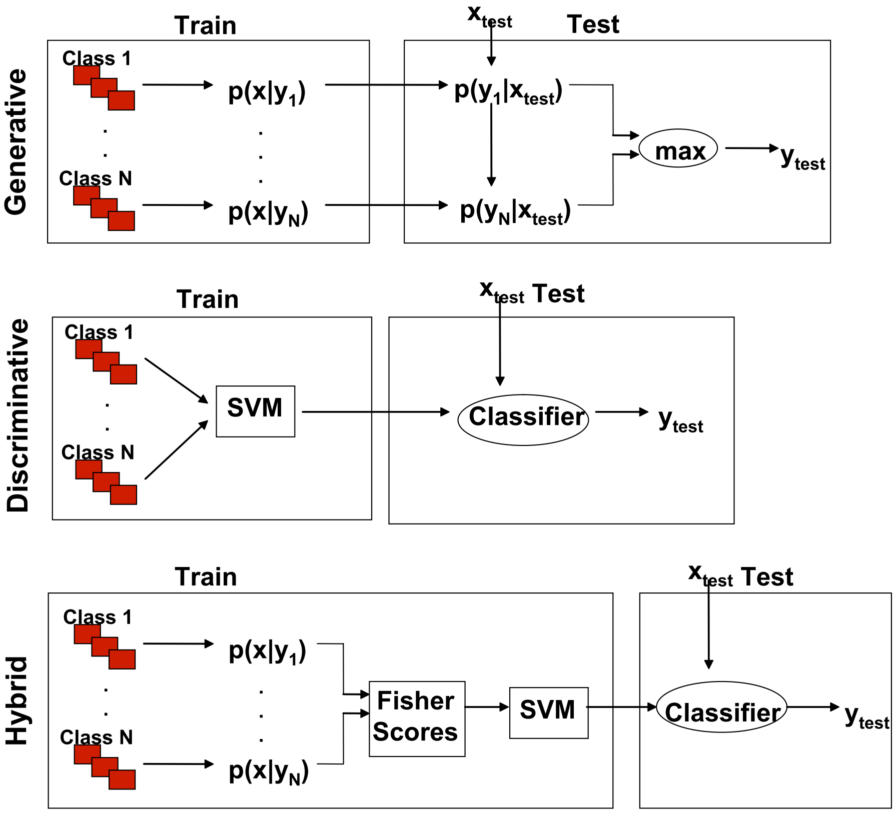

Fig. 2 Schematic comparison of the generative (*top*), discriminative (*middle*), and hybrid (*bottom*) approaches to learning discussed in this paper. While generative models are a natural choice for visual recognition, discriminative models have been shown to give better performance in different domains. The hybrid model captures many desirable properties of both

ing worse. Third, it has been shown that prior knowledge can be useful when few training examples are available, and that prior information may be easily incorporated in a generative model (Fei-Fei et al. 2004). Fourth, we ultimately envision systems which can learn thousands of categories; in this regime it is unlikely that we will be able to learn discriminative classifiers by considering simultaneously all the training data. It is therefore highly desirable to design classifiers that can learn one category at a time: this is easy in the generative setting and difficult in the discriminative setting where training data for all categories must be available at once for a decision boundary to be calculated. Fifth, it is unclear, in general, what features to use when training a discriminative classifier on object categories. Consider that many popular algorithms for object recognition rely on feature detectors to find 'interesting' regions within an image. Each image thus is represented as an unordered set of feature detections of variable length. How can these unordered lists be used by a discriminative classifier?

Is it possible to get the best of both worlds and develop approaches with the flexibility of generative learning and the performance of discriminative methods? Jaakkola and Haussler have shown that a generative model can be used in a discriminative context by extracting Fisher Scores from the generative model and converting them into a 'Fisher Kernel' (Jaakkola et al. 1999) (see Fig. 2). A kernel represents the data as a matrix of pairwise similarities which may be used for classification by a kernel method, such as the support vector machine (SVM). The field of kernel methods is well developed (Vapnik 1998; Shawe-Taylor and Cristianini 2004; Schoelkopf and Smola 2002) and represents the state-of-the-art in discriminative learning. Here, we explore how to apply these ideas to visual recognition.

We calculate Fisher Kernels that are applicable to visual recognition of object categories and explore experimentally the properties of such 'hybrid models' on a number of popular and challenging data-sets. Other kernel-based approaches have been suggested for object recognition, including Vasconcelos et al. (2004) who exploit a similar paradigm, using a Kullback–Leibler based kernel and test on the COIL data-set. Wallraven et al. (2003) utilize a clever kernel which implicitly compares detected features in different images, but apply their method to different sets of images than those used in this paper.

In Sect. 2 we briefly review one class of generative models, commonly called the 'Constellation Model', which will be used in the rest of the paper. In Sect. 3 we show how to transform a generative Constellation Model into a discriminative setting by utilizing the idea of Fisher Kernels. In Sect. 4 we compare the performance of hybrid and generative constellation models. In Sect. 5 we explore how these hybrid models can be extended and effectively used in circumstances where we have a mixture of labeled and unlabeled data, i.e. 'semi-supervised' learning. Finally, in Sects. 6 and 7 we show how the hybrid framework can be used to optimally combine several generative models (for example generative models based on different feature detectors and different numbers of parts) into a single classifier. Section 8 discusses the main results and observations of this work.

2 Generative Models

In this section we briefly review a class of generative models which will be used in conjunction with the discriminative methods described in the next section. In principle, any generative model that is differentiable with respect to its parameters can be used. We chose to experiment with the 'Constellation Model' which was first proposed by Burl and Perona (1996). Weber et al. (2000) showed that this model may be learned from cluttered images in a weakly supervised setting in which only a class label is associated with each image using maximum likelihood. Fergus et al. (2003) extended the model by making it scale-invariant and incorporating general purpose feature detectors. We use a simplified version of Fergus' constellation model in which we do not explicitly model occlusion or relative scale.

2.1 The Constellation Model

The constellation model is a generative framework which constructs probabilistic models of object classes by representing the appearance and relative position of several object parts (Burl and Perona 1996; Weber et al. 2000; Fergus et al. 2003). Given a suitable training set composed of images containing examples of objects belonging to a given category, such models are trained by finding a set of model parameters θ^{MLE} which maximizes the log-likelihood of the model (Weber et al. 2000; Fergus et al. 2003). Both appearance and shape are modeled as jointly Gaussian and $\theta = \{\theta_a, \theta_s\}$ represent the mean and diagonal variance parameters of the shape (θ_s) and appearance models (θ_a). To remove dependence on location, the x-y coordinates of the parts are measured relative to a reference part, e.g. the leftmost part. In our implementation, as suggested by Fergus et al. (2003), appearance is represented by the first 20 PCA components of normalized 11×11 pixel patches which were cut out around feature detections in training images at the scale indicated by the detectors (see next subsection). The number of interest point detections considered in an image is a design parameter.

For each training image I_i we obtain a set of F interest points and their appearance descriptors. We would like to establish correspondence, i.e. assign a unique interest point to every model part, or component, M_j. Burl and Perona (1996) showed that since we do not *a priori* know which interest point belongs to which model component, we need to introduce a 'hidden' hypothesis variable h which maps interest points to model parts. We order the interest points in ascending order of x-position. Note that although we model only the diagonal components of the Gaussian, the model parts are not independent as we enforce that each part is mapped to a unique feature, implicitly introducing dependencies. The result is a total of $\binom{F}{M}$ hypotheses, where each h assigns a unique interest point to each model part. We marginalize over the hypothesis variable to obtain the following expression for the log likelihood for a particular class:

$$\sum_i \log(p(I_i)) = \sum_i \log\left(\sum_h p(A_i, h|\theta_a)p(X_i, h|\theta_s)\right) \quad (1)$$

where $\{X_i\}$ are the relative coordinates of the object and represent the shape information while $\{A_i\}$ are the PCA components described above and represent the appearance information. We assume that the shape and appearance models are independent of one another given a hypothesis h and that the images are I.I.D. This step is key to maximum-likelihood model learning, and to classification, once a model is available (see details in Burl and Perona 1996; Weber et al. 2000; Fergus et al. 2003). We note that exploring all possible hypotheses carries a combinatorial computational cost which severely limits the number of parts and interest points which can be used in the model.

For clarity we consider the makeup of a typical set of parameters θ. Consider one part of a 3-part model. A single part consists of parameters specifying its shape and appearance, θ_s and θ_a respectively. The shape of the part is specified by a two dimensional mean and two dimensional variance (we consider diagonal covariance matrices in our model) indicating the mean and variance of the position of the part. Each part thus has a four dimensional parameter array specifying its location. Now consider the appearance parameters of a part. The appearance of a part is specified by the mean and variance of the PCA components for that particular part. A 20 dimensional PCA representation thus consists of a total of 40 parameters, 20 for the mean and 20 for the variance of the part.

2.2 Interest-Point Detection

The constellation model requires the detection of interest points within an image. Numerous algorithms exist for ex-

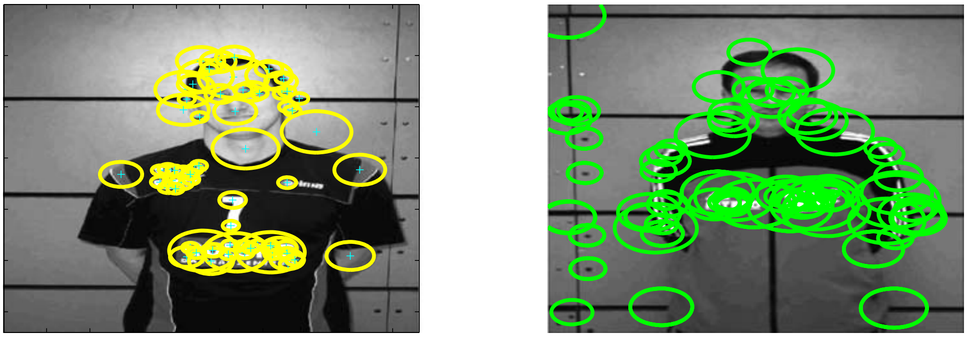

Fig. 3 Examples of scaled features found by the KB (left) and multi-scale DoG (right) detectors on images from the 'persons' data-set located at http://www.emt.tugraz.at/opelt/. Approximately the top 50 most salient detections are shown for both

tracting and representing these interest points. We considered several popular interest point detectors: the entropy based Kadir and Brady (KB) (Kadir and Brady 2001) detector, the multi-scale Difference of Gaussian (DoG) detector (Crowley 1984), the multi-scale Hessian detector (mHes), and the multi-scale Harris detector (mHar). Figure 3 shows typical interest points found within images. All detectors indicate the saliency of interest points, and only the most salient interest points are used. The KB interest point detector was used in all experiments below unless otherwise specifically noted.

2.3 Generative Model Learning

We train our generative constellation models using the EM algorithm (Dempster et al. 1977) as computed explicitly for the constellation model by Weber et al. (2000). The algorithm involves iteratively calculating the expected values of the unobserved variables of the model and then maximizing the parameters. The algorithm was terminated after 50 iterations or after the log likelihood stopped increasing. A typical 3-part model optimized on 100 images with 25 detections in each image took on the order of 20 minutes to optimize using a combination of C (mex) and Matlab code on a 2 GHz machine.

3 Fisher Scores and Fisher Kernels

For supervised learning, such as regression and classification, kernel methods are often the method of choice. As argued in the introduction, our interest is in *combining* generative models with a discriminative step for the purpose of visual object recognition. We chose support vector machines (SVM) (Vapnik 1998) as our kernel machine. The SVM (like all kernel methods) process the data in the form of a kernel matrix (or Gram matrix), a symmetric and positive definite $n \times n$ matrix of similarities between all samples. A simple way to construct a valid kernel matrix is by defining a set of features, $\boldsymbol{\phi}(x_i)$, and to define the kernel matrix as,

$$K_{i,j} = K(x_i, x_j) = \boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x_j). \tag{2}$$

The kernel represents the similarities between samples: relatively large kernel entries correspond to two samples which are similar while small (possibly negative) entries correspond to dissimilar samples. Kernels defined by inner products such as the one in (2) produce positive-definite kernel matrices (Vapnik 1998).

The generative model will have its impact on the classifier through the definition of these features. We will follow (Jaakkola et al. 1999) in using 'Fisher Scores' as our features. Given a generative probabilistic model the 'Fisher Scores' $\phi(x_i)$ are defined as

$$\boldsymbol{\phi}(x_i) = \frac{\partial}{\partial \boldsymbol{\theta}} \log p(x_i|\boldsymbol{\theta}^{\mathrm{MLE}}) \tag{3}$$

where $\boldsymbol{\theta}^{\mathrm{MLE}}$ is the maximum likelihood estimate of the parameters $\boldsymbol{\theta}$. By definition, $\boldsymbol{\theta}^{\mathrm{MLE}}$ is obtained by maximizing the likelihood. A necessary condition is that the gradient of such likelihood (or log-likelihood) is zero, which is equivalent to 'balancing' the Fisher Scores,

$$\sum_i \frac{\partial}{\partial \boldsymbol{\theta}} \log p(x_i|\boldsymbol{\theta}^{\mathrm{MLE}}) = \sum_i \boldsymbol{\phi}(x_i) = 0. \tag{4}$$

Hence, samples "pull" on the parameter values through their Fisher Scores which can be interpreted as "forces". At the MLE all forces balance. Two data-items that exert similar 'forces' on all parameters have their feature vectors aligned resulting in a larger positive entry in the kernel matrix.

Since it is not *a priori* evident that the data can be separated using a hyperplane in this feature space, it can be beneficial to increase the flexibility of the separating surface (making sure that the problem is properly regularized) as shown in (Vapnik 1998). This is achieved by applying non-linear kernels such as the RBF kernel or the polynomial kernel in this new feature space, i.e. $K(\boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j))$ with,

$$K_{\mathrm{RBF}}(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}||\boldsymbol{\phi}(x_i) - \boldsymbol{\phi}(x_j)||^2\right), \tag{5}$$

$$K_{\mathrm{POL}_p}(x_i, x_j) = (R + \boldsymbol{\phi}(x_i)^T \boldsymbol{\phi}(x_j))^p \tag{6}$$

where σ represents the variance of the RBF kernel and p represents the degree of the polynomial kernel being used.

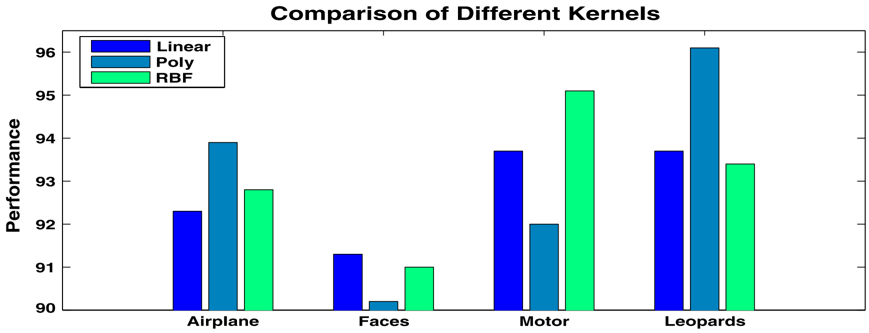

Fig. 4 Performance comparison of various kernels on several datasets. The parameters used to train and test these models are described in the experimental section. The polynomial kernel was of degree 2. The y-axis indicates the classification performance, note that the scale starts at 90%. These results were averaged over 5 experiments. 100 train/test examples used. *First bar in each set*: Linear kernel. *Second bar*: Polynomial kernel. *Third bar*: RBF kernel

To remove scale differences between the features we normalized the features before we computed their inner product,

$$\phi_a(x_i) \to \frac{\phi_a(x_i)}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}\phi_a^2(x_j)}}. \tag{7}$$

Why do we bother going through a two-stage process where we first train generative models for each object category and then train another classifier based on a kernel derived from those models, where we could also classify using log-likelihood ratios? The intuitive answer to this question is that a classifier is trained to find an optimal decision boundary, i.e. it focuses its attention to what is relevant to the task. Here, the samples which are close to the decision boundary carry much more information than the ones away from the boundary. In contrast, classifying according to likelihood ratios simply derives the decision boundary as a byproduct from fitting models for every category. The objective of this fitting procedure is to maximize the probability of all samples for every category and not deriving a good decision boundary for the classification task at hand. This intuition has been made more precise in numerous papers. Most relevant to Fisher Kernels is the theorem in (Jaakkola et al. 1999) stating that asymptotically (in the large datalimit) a classifier based on the Fisher Kernel can be shown to be at least as good (and typically better) as the corresponding naive Bayesian procedure (i.e. likelihood ratios or maximizing $p(y|x)$). Similar results have been obtained in e.g. Ng and Jordan (2002) and Tsuda et al. (2003). It should be mentioned that for small numbers of samples the naive Bayesian procedure may act as a regularizer and avoids the kind of over-fitting that can be observed in discriminative approaches.

Given a kernel matrix and a set of labels $\{y_i\}$ for each sample, the SVM proceeds to learn a classifier of the form,

$$y(x) = \text{sign}\left(\sum_i \alpha_i y_i K(x_i, x)\right) \tag{8}$$

where the coefficients $\{\alpha_i\}$ are determined by solving a constrained quadratic program which aims to maximize the margin between the classes. For details we refer to (Shawe-Taylor and Cristianini 2004) and (Schoelkopf and Smola 2002). In our experiments we used the LIBSVM package (available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/).

There are a number of design parameters: the free parameters in the definition of the kernel (i.e. σ in the RBF kernel and R, p in the polynomial kernel) and some regularization parameters in the optimization procedure of the $\{\alpha_i\}$. For an SVM the regularization parameter is a constant C determining the tolerance to misclassified samples in the training set. Values for all design parameters were obtained by cross-validation or by learning them on a bound of the leave-one-out error (see Sect. 6).

In Fig. 4 we compare the performance of the various kernels defined above on two data-sets. The performance is similar, with some variability between data sets. We used linear and RBF kernels in the following experiments as there was no appreciable difference in the performance of the various kernels and because these kernels are popular within the machine learning community.

3.1 Fisher Scores for the Constellation Model

In order to train an SVM we require the computation of the Fisher Score for a model. Recall that the Fisher Score is the derivative of log likelihood of the parameters for the model, i.e.:

$$\frac{\partial}{\partial\theta_s}\log(p(I_i|\theta)) = \sum_h p(h|I_i,\theta)\frac{\partial}{\partial\theta_s}\log p(X_i,h|\theta_s), \tag{9}$$

$$\frac{\partial}{\partial\theta_a}\log(p(I_i|\theta)) = \sum_h p(h|I_i,\theta)\frac{\partial}{\partial\theta_a}\log p(A_i,h|\theta_a) \tag{10}$$

where both $\{\theta_a, \theta_s\}$ consist of mean and variance parameters of Gaussian appearance and shape models. Despite a potentially variable number of detections in each image I_i its Fisher Score has a fixed length. This is because the hypothesis h maps features to a pre-specified number of parts and hence there is a fixed number of parameters in the model.

Most of the execution time of the algorithm is spent during the computation of the Fisher Kernels as well as the training of the generative models. In comparison, the SVM training, even with extensive cross-validation, is quite short due to the relatively small number of training images.

4 Comparing Generative and Hybrid Approaches

Our first set of experiments was designed to compare the performance of our hybrid method with generative models on a commonly used benchmark of 4 object categories with diverse appearances from one another. We chose the same categories used by Fergus et al. (2003) in order to directly compare with their results (their results were obtained using a more sophisticated generative Constellation Model than the one we used).

Some details of the SVM training: Fisher Scores were normalized to be within the range $[-1, 1]$. We performed 10x cross-validation to obtain estimates for the optimal values of C. For the RBF kernel, which contains the additional hyper-parameter σ, we found the optimal values of C and σ by performing an exhaustive search over the parameters. We varied the cross-validation search space for both parameters on a log base 2 scale, $C \in 2^{[-7,9]}$ and $\sigma \in 2^{[-8,0]}$. We chose this range because the best results were typically within these parameter settings.

4.1 Experiments on Caltech Data-Sets

Table 1 illustrates the performance on these data-sets.[1] Images were normalized to have the same number of pixels in the x-dimension in order to prevent the algorithm from learning meaningful information from the absolute size of the images. We did not crop the images. We recognize that although these data-sets are used extensively by the vision community, they exhibit some deficiencies. In particular the object of interest is usually in the center of the image and the objects are mostly in standardized poses and good lighting conditions. In these experiments a single generative model was created of the foreground class from which Fisher Scores were extracted for both the foreground and background classes for training. The SVM was trained with the Fisher Scores from the foreground and background class. Testing was performed using an independent set of images from the foreground and background class by extracting their Fisher Scores from the foreground generative model and classifying them using the SVM. We refer to these experiments as 'class vs. background' experiments, as they involve a discrimination task between one foreground class and one background class.

Table 1 Performance comparison for some Caltech data-sets in a 2-alternative task whether the test image either contains an object from a given class, or contains no object (background class). We used 100 training and test images for each class although performance did not increase significantly when more training images were used. The background class was the same used by Fergus (2005). All scores quoted are the total number correct for both the target class and the background over the total number of examples from both classes. The second column shows the performance of our hybrid discriminative algorithm. The third and fourth columns show performance using only the Shape and Appearance Fisher Scores respectively. The Fifth column is the performance using a likelihood ratio on the underlying generative models. The final column shows previous performances on the same data-sets (Fergus 2005). Our underlying generative model contained 3 parts and used a maximum of 30 detected interest points per image. Results were averaged over 5 experiments. In comparing with Fergus (2005) note that in that study approximately twice as many training images were used, as well as a more sophisticated generative constellation model (6 parts, scale-invariance, occlusion modeling), hence the higher performance of (Fergus 2005) with respect to our baseline generative constellation model (ML). On the other hand, our hybrid method models relies on the background images for SVM training while the ML method of (Fergus 2005) does not make explicit use of the background images. Test performance did not vary significantly between different experiments

Category	Hybrid	Shape	App	ML	Prev
Faces	**91**	77.7	88.9	83	89.4 (Fergus 2005)
Motorcycles	**95.1**	74.5	91.2	74.2	96.7 (Fergus 2005)
Airplanes	**93.8**	95.3	84.2	72.4	92.2 (Fergus 2005)
Leopards	**93**	71.8	91.3	68.1	88 (Fergus 2005)

In addition to class vs. background experiments, we conducted classification experiments using multiple object categories. First a generative model was constructed for all classes of interest. Fisher Scores for both train and test images were obtained by concatenating the Fisher Scores from each model. Only the training images were used to create both the SVM classifier and the generative distribution. Since an SVM is inherently a two-class classifier we train the multi-class SVM classifier in a 'one-vs-one' manner. For each pair of classes a distinct classifier was trained. A test image was assigned to the category containing the largest number of votes among the trained classifiers. 'One-vs-one' classification requires that $\frac{N(N-1)}{2}$ classifiers be created and used during classification. 'One-vs-one' clearly introduces a large number of classifiers which must be evaluated during run-time. However, we prefer it to a 'one-vs-all' strategy as 'one-vs-all' is not as amenable to unbalanced data-sets and the potential large computational cost is not as evident for small numbers of training examples. All other parameters

[1] The data-sets, including the background data-set used here, can be found at: http://www.vision.caltech.edu/html-files/archive.html. The Leopards data-set is from the Corel Data-Base.

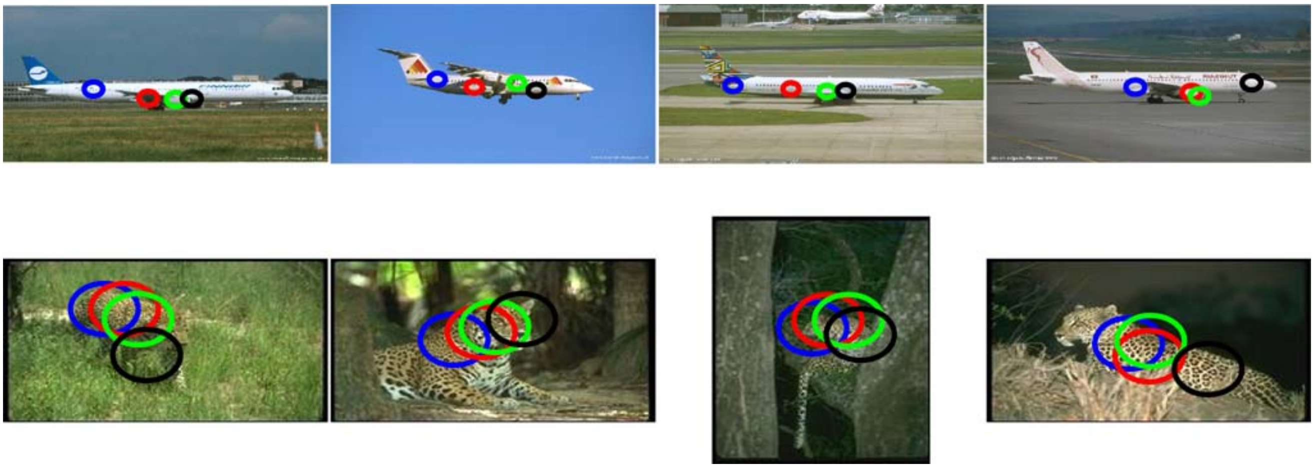

Fig. 5 Localization of objects within images using the generative constellation model. Each unique colored circle represents a different part of the model. This is a 4-part model. The positions of the circles represent the hypothesis, h, with the highest likelihood. The generative framework approximately localizes the position of each object. We note that the images shown here do not exhibit excessive amounts of clutter

Table 2 Confusion table for 4 Caltech data-sets in a 4-way classification experiment. The main diagonal contains the percent correct for each category. Perfect performance would be indicated by 100 s along the main diagonal. We use the same classes as used in (Fergus et al. 2003) which utilizes a purely generative constellation model and resulted in performances of 92.5, 90.0, 96.4, and 90.2 across the main diagonal for a 4-way discrimination task for an average performance of 92% while we achieve an average performance of 96%. When performing classification using only the Shape and Appearance Fisher Scores we achieve an average performance (average across the main diagonal of the confusion table) of 72.6 and 94.8. 100 training and testing images were used in our experiments. Averaged over 3 experiments. Test performance did not vary significantly between different experiments

True Class ⇒	Motor	Leopards	Faces	Airplanes
Motorcycles	**96.7**	7.3	1.3	0.3
Leopards	1.3	**90.7**	0.7	0
Faces	1.3	0	**97**	0.3
Airplanes	0.7	2	1	**99.3**

for training were kept the same as above. Table 2 illustrates a confusion table for 4 Caltech Data-Sets. The discriminative method again outperforms its generative counterpart (Fergus et al. 2003) despite using a much simpler underlying generative model.

There are several interesting points of note. First, the hybrid method works well even on dissimilar categories. In fact it performs significantly better than the corresponding generative ML method and slightly better than a more sophisticated ML method. Second: the classification results in Table 2 are comparable with the best current results in the literature. It is fair to say that the Caltech-4 data-set is easy (see Figs. 5 and 6) and may not be the best data-set to use when comparing algorithms. Further results on more challenging data-sets, among them the 'People' and 'Bikes' Graz data-sets,[2] were reported by us in (Holub et al. 2005) and the technique performed well in these cases as well. Three, Experiments conducted using only the Shape and Appearance Fisher Scores mostly indicate that the combination of the two is more powerful than either in isolation. Interestingly, by comparing the shape-only and appearance-only performance one can see that the relative importance of these two terms varies with the category. In particular we notice that the Leopard data-set, which exhibits stereotyped appearance information but large articulations in shape, performs best when using only appearance information while airplanes, which exhibits fairly uniform shape information, yields good performance when using only shape information.

The hybrid approach uses an underlying generative constellation model to generate Fisher Scores which are in turn used for classification. This underlying model can be used to localize objects within an image by selecting the hypothesis with the highest likelihood (the same technique was demonstrated to localize objects by in Fergus et al. (2003)). Figure 5 demonstrates the localization ability of our implementation of the constellation model on several different categories.

4.2 Background Classes

In this section we explore the effect of a particular "background" training set on learning. Consider the detection tasks described above (Fig. 1) in which we build a hybrid

[2] Available at http://www.emt.tugraz.at/ opelt/.

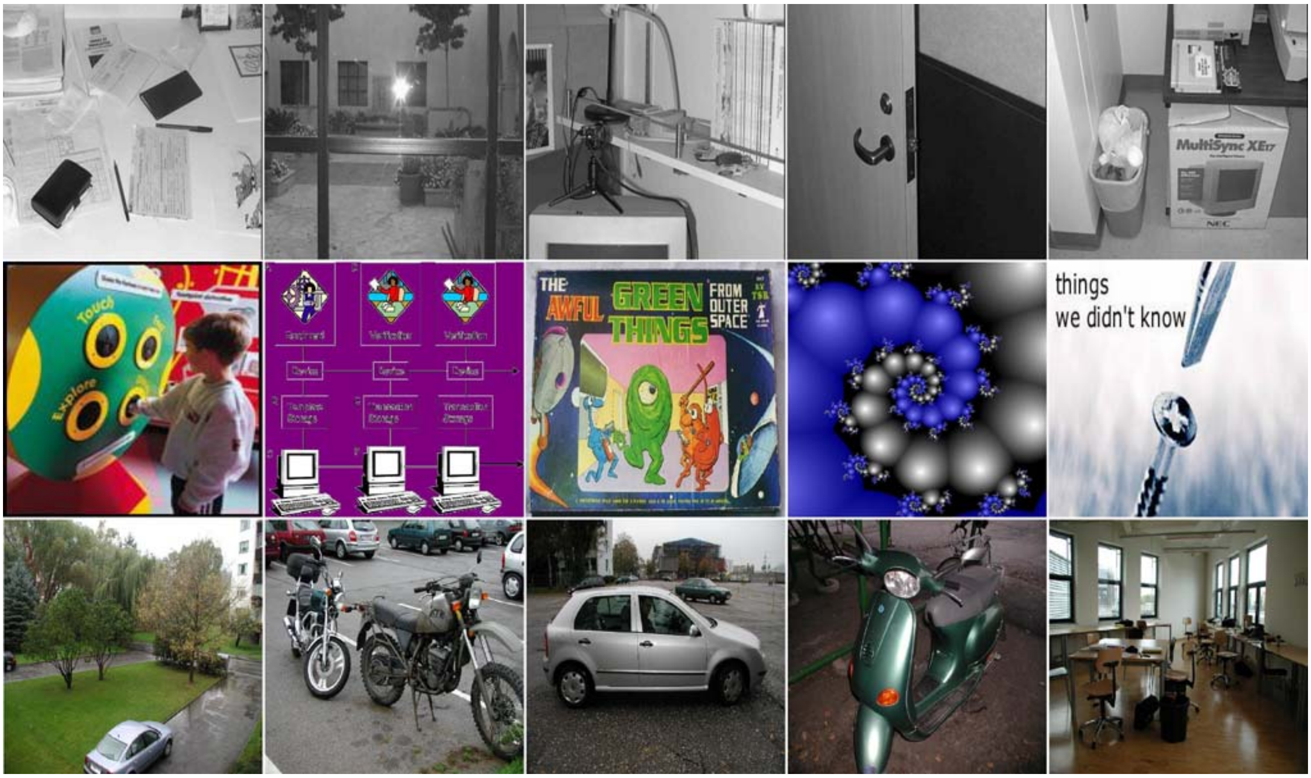

Fig. 6 Examples of background images: (*top*) Caltech 1 is a collection of indoor and outdoor digital photographs taken on the Caltech campus, (*middle*) Caltech 2 is a collection of images obtained from the Google image search engine by typing 'things' as a search string, (*bottom*) Graz is a collection of outdoor and indoor images, some of which focus on specific objects. There are noticeable differences in the image statistics from the different background classes

classifier which detects whether or not an object is present in an image. In this setting, the algorithm must be trained using both a 'positive' or foreground training set and a negative or 'background' training set. In principle the background set should represent any images that does not contain the object of interest. This set of images has a very broad statistical variation which may not be well represented by a limited training set as used in our experiments. If the statistics of the background data-set are not appropriately chosen, we run the risk of over-fitting to those background images used for training.

In order to further explore this potential confound, we performed experiments using several common data-sets used as background images to determine how well background data-sets generalized to one another. We considered 3 standard sets of background images: (1) the Caltech background data-set used in (Fergus et al. 2003) (Caltech1), (2) the Caltech background data-set used (Fei-Fei et al. 2004) (Caltech2), (3) the Graz data-set used in (Opelt et al. 2004) (Graz). A random sample of images from the three sets is shown in Fig. 6. We performed experiments by first generating a model for one foreground class. This generative model was then used to create Fisher Scores for the foreground class and a particular background class and an SVM classifier was trained using these scores. We *tested* the classifier on images from all three background classes.

Results for these experiments are summarized in Table 3. This table illustrates that the statistics of a particular background data-set can influence the ability of the classifier to generalize to new sets. These results should be seen as a caveat when using discriminative learning for detection tasks as the statistics of the background images play a crucial role in generalization, especially when relatively few background examples are used. Even if one has access to a large number of background images it is in principle, difficult, to obtain a set of images which model the distribution of any arbitrary background. Furthermore, some discriminative methods, such as SVMs, are not readily amenable to training with unbalanced data-sets, making the choice of background images to use during training even more problematic.

5 Semi-Supervised Learning

In computer vision it is often easy to obtain unlabeled images while labeled images often require a significant investment of resources. In this section we explore how to leverage

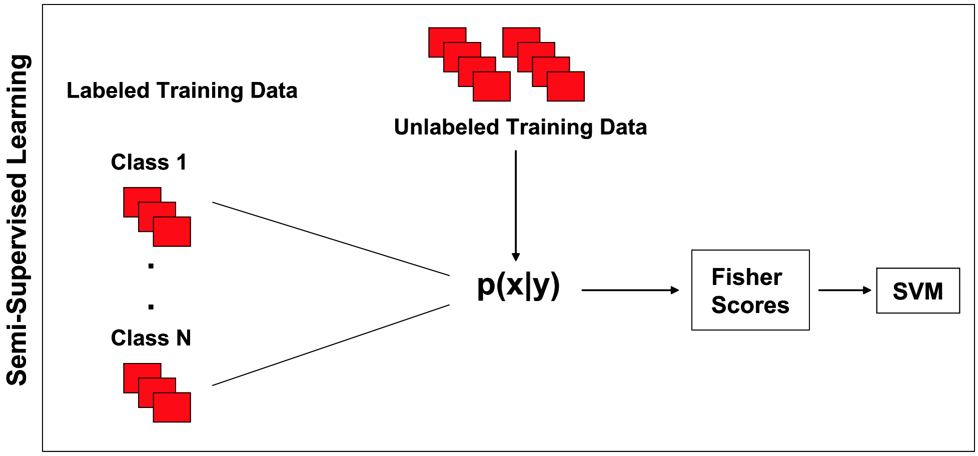

Fig. 7 Schematic overview of the semi-supervised learning algorithm employed. Semi-supervised learning employs both labeled and unlabeled data. Note that only unlabeled data is used to create the generative model but that Fisher Scores are extracted from the generative model using labeled data

Table 3 Generalization of different background statistics: Top, Airplane vs. BG experiments. Bottom, Leopards vs. BG experiments. The top row indicates the background data-set trained with. The rows indicate the test set used. The columns indicate the performance of the algorithms. The bold scores indicate the performance on the test examples from the same background class which was trained on, these tend to be the highest performing test sets. We trained and tested with 100 images for each foreground and background category. Results were averaged over 2 experiments. The mHar detector was used

Trained BG ⇒	Caltech1	Caltech2	Graz
Caltech1	**93**	86.5	82
Caltech2	83	**90.5**	78
Graz	83.5	88	**91**
Caltech1	**92**	82	83.5
Caltech2	83	**92.5**	87
Graz	85	85	**91.5**

unlabeled and labeled images within our hybrid generative-discriminative framework described above. Using labeled and unlabeled data is often referred to as *semi-supervised learning* in the machine learning community. In particular we show how semi-supervised learning can be used to learn classifiers with far fewer training examples than the corresponding supervised framework. Figure 7 illustrates a schematic of the proposed semi-supervised learning algorithm.

Many interesting methods have been proposed for semi-supervised learning (see e.g. Seeger 2002 for an approach relevant to Fisher Kernels). We decided to implement a relatively simple idea that attempts to learn the kernel using the available unlabeled data. In the context of Fisher Kernels this boils down to learning a probabilistic model on the unlabeled data-set. We subsequently extract Fisher Scores based on this model, but evaluate it on the *labeled* data. These Fisher Scores are combined into a kernel matrix and provide input to the SVM. To see why this is a sensible approach we consider the task of classifying images of faces of two different people. The Fisher Scores represent the derivatives of the log-likelihood and hence the tendency of a particular sample to change the model. If the underlying model is one of a completely unrelated class of objects (say leopards), then we expect any face image to roughly change the model in a similar fashion, so the Fisher Scores for different faces are expected to be similar. If however, the underlying model is an average face model, then we expect the changes to the model for person A and person B to be different, resulting in different Fisher Scores and hence small kernel entries. So clearly, a good kernel should be based on a probabilistic model for the class of objects we are trying to classify.

In the following sections will investigate a number of issues: (1) does the method we propose work at all, (2) what is the effect of using different sets of unlabeled data, (3) how does the performance depend on the number of unlabeled examples and (4) how does the performance depend on the number of labeled (training) examples.

5.1 Caltech Faces-Easy Categories

We employ a similar experimental paradigm as described above in Sect. 4. However, we consider situations when there is a wealth of unlabeled data available which will be used to construct the model from which we extract Fisher Scores. In order to be clear on our terminology we refer to the 'unlabeled training set' as the set of unlabeled data used to generate the classifier and the 'labeled training set' as the set of labeled data used to generate the classifier.

The Caltech Faces-Easy (see Fig. 8) consists of about 400 images of human faces. It is composed of 20 or more pho-

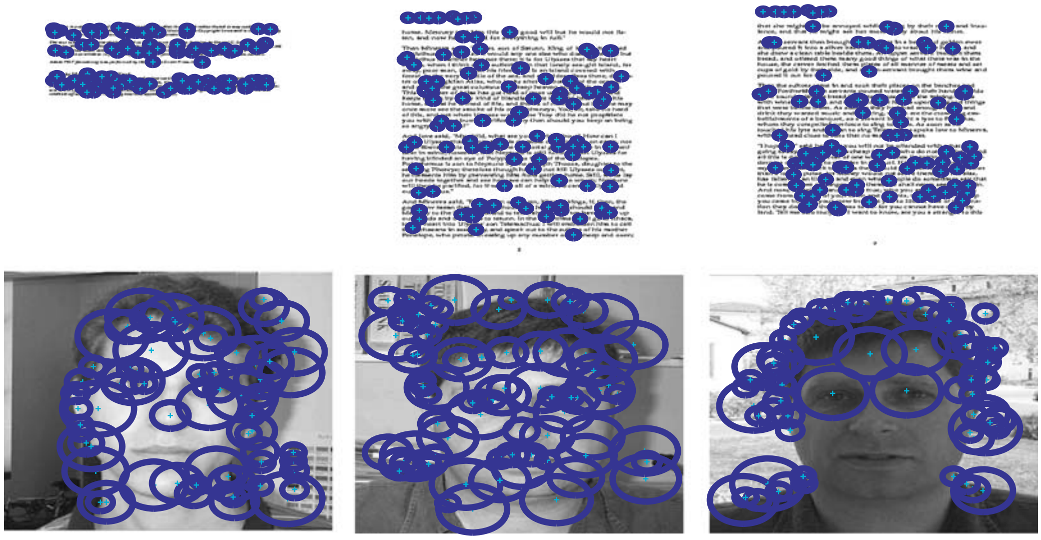

Fig. 8 Features detected by the Kadir and Brady detector on image of (*Top*) the Odyssey text and (*Bottom*) the Faces-Easy data-set. Note that the Faces-Easy data-set is equivalent to the Faces data-set used in the experiments above except that the Faces-Easy data-set contains faces which are more cropped. Also note that the features found in the two sets of images have drastically different appearance statistics

tographs of 19 individuals, while the remaining 40 or so images contain fewer exemplars. Thus we can divide the entire face category into smaller categories corresponding to individuals.

A linear kernel was used in all experiments below. Using an exponential kernel is difficult due to the inability to accurately tune the scale parameter σ: there are not enough exemplars to perform cross-validation.

5.2 Results

We compared performance using various different sets of unlabeled training images to create the generative model, where each set of images was more or less related to the set of labeled training images. We considered the following sets of unlabeled training images: (1) Faces-Easy data-set. Note that our unlabeled training set contained images from all individuals except those used for the labeled training set. (2) the Caltech 1 Background Images (see Table 3 above), (3) the Leopards data-set, (4) Images of printed pages from a copy of Homer's 'Odyssey'. Examples of the features found for classes (1) and (4) are shown in Fig. 8. A generative model trained on background will result in broad distributions, while generative models trained on specific classes will have a more distinct distribution reflecting the statistics of the unlabeled training images.

First a model was trained using images from the unlabeled training data-sets. Next, 2 individuals from the Faces-Easy set were selected at random to train the classifier. Fisher Scores were extracted from the model. We did not include any images used for training or testing in the unlabeled data-set. Median and 25th/75th quantiles were computed over 20 experiments by selecting to individuals at random.

Results are shown in Fig. 9. Several interesting points can be made: (1) The nature of the unlabeled training data-set is critical: using a data-set unrelated to the classification problem at hand, e.g. the 'The Odyssey' or 'Leopards' data-sets in the context of face classification, results in the worst performance. Using a very general data-set, e.g. the Caltech 1 Background (BG1) data-set, results in better performance than using an unrelated data-set. This may be due to the broad distribution which results from training on the background data-set, which, although not as beneficial as using the images from the same class, is better than using a generative model from a completely different class. Finally, using a data-set which describes the distribution of the images involved in the classification problem well, e.g. using the remaining face images from "Easy-Faces" to classify the faces of two held-out individuals, results in the best performance. (2) Discrimination performance increases as we add more unlabeled training examples. This is particularly true for the faces data-set, in which we notice a large increase in performance from 10 to 100 prior examples. The same qualitative effect is observed for the BG1 data-set. We observed that with few training examples the BG1 unlabeled images creates overfitted models with small variances, resulting in comparable performance to using the 'Odyssey' or 'Leopards' data-sets. As more training examples are added, the model becomes more general, and its utility as a prior improves.

A reasonable measure for how appropriate a kernel is for a particular classification task is the "kernel-alignment" pro-

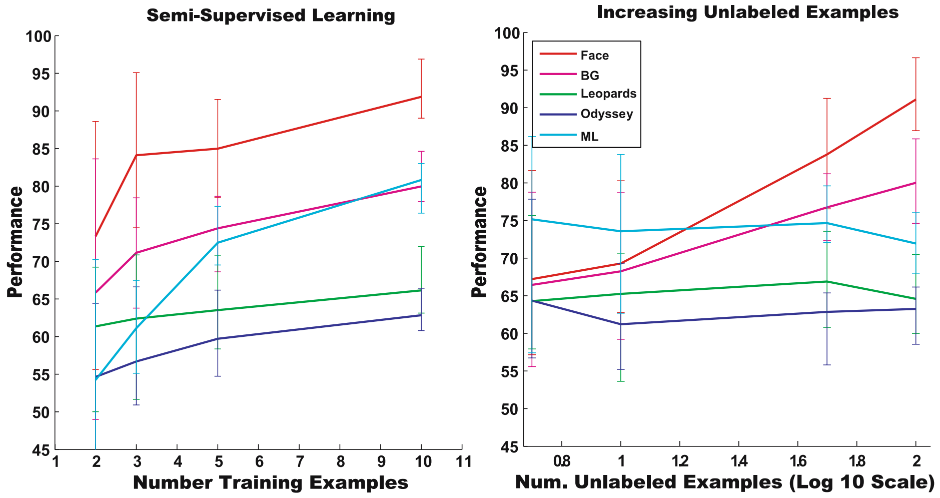

Fig. 9 (*Left*) Performance on the 2-way faces classification problem as we vary the number of labeled training examples per class. Number of unlabeled examples was fixed at 100. (*Right*) Classification performance as a function of the number of unlabeled examples. 5 labeled training examples per class were used to train the SVM classifier. In both plots we show the median and 25th/75th quantiles computed over 20 experiments. Each line represents a different unlabeled data-set except for the cyan line which shows the maximum likelihood performance on the labeled training set only (i.e. it did not use any unlabeled data and classified by comparing the likelihood scores, the fluctuations in this line are due to using randomized training/test sets). Note that the nature of the unlabeled data-set has an important impact on the classification performance. Experiments using 5 training examples on the left plot correspond to the experiments using 100 unlabeled examples on the *right plot*

posed in (Shawe-Taylor and Cristianini 2004),

$$A(\mathbf{y}\mathbf{y}^T, K) = \frac{\mathbf{y}^T K \mathbf{y}}{N\sqrt{\mathbf{tr}(K^T K)}} \tag{11}$$

where N is the number of training cases and $\mathbf{y}$ is a vector of $+1$ and -1 indicating the class of each data-case.[3] Figures 10 and 11 illustrate that training using more suitable sets of unlabeled data results in more appropriate kernels.

6 Combining Multiple Generative Models

In the previous section we showed how to leverage unlabeled data within our hybrid framework. In this section we show how the hybrid framework can be used to combine multiple generative models into a single classifier. Why is this useful? Consider that visual data is heterogeneous in nature. Different 'front-ends' (feature detectors, feature descriptors) work best on different types of data. For instance, the techniques used for optical character recognition (OCR) vary greatly from those used to detect cars. Also, it is clear that some features should be described with 'brightness' templates, others with 'texture' descriptors, still others with parameterized edges or curves. It would be useful to remain initially agnostic as to which front-end is most useful and allow the learning algorithm to decide which to use.

In the previous section we learned the kernel on unlabeled training data under the assumption that very few labeled training data were available. However, it is also possible to tune the kernel based on the labeled training examples, provided the labeled data-set is sufficiently large. In this setting, one has to be careful not to "overfit" the kernel on the training examples and achieve poor generalization performance. A standard approach to determine regularization and kernel parameters is by cross-validation. Unfortunately, when the number of parameters is large this method becomes infeasible. An alternative approach that has been proposed in the literature is to optimize the kernel on approximations or bounds of the test-error. The approach we will follow here was described in Chapelle et al. (2002) (see also Opper and Winther 2000 and Jaakkola and Haussler 1999) which derive gradients for the span bound of the leave-one-out error. Other authors, e.g. Gold et al. (2005) offer an alternate solution to this problem.

Consider the choices we face when modeling a particular object category using the Constellation Model. Which front-end interest point detector should we use? How many parts should the model contain? Should one model the geometry and appearance of a part? Each of these choices leads to a different model and hence to a different set of Fisher Scores. Instead of choosing a particular type of model, one could argue to create multiple models, and incorporate information

[3] Why did we not choose the perfect kernel according to this metric, namely $K = \mathbf{y}\mathbf{y}^T$? The reason is that this kernel would overfit on the training data and exhibit poor generalization performance. The alignment measure is therefore only useful provided we do not overfit, which we would not expect given the fact that we learn the kernel on a separate, unlabeled data-set.

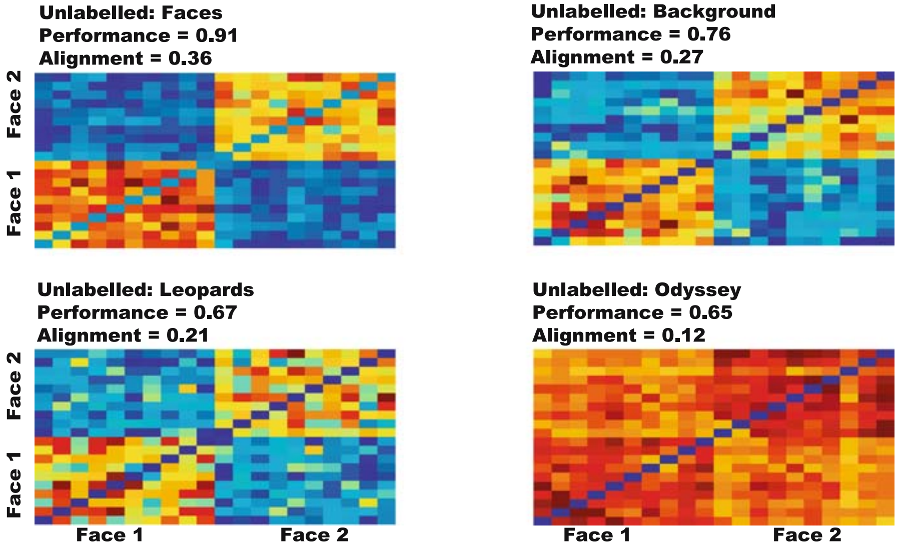

Fig. 10 Kernel matrices computed using different sets of unlabeled training data. Each model underlying a kernel was trained on 200 unlabeled data-cases. A kernel was computed as $\boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x_j)$ with normalized Fisher Scores and averaged over 20 experiments. Diagonal entries are zeroed out to improve resolution. When the Faces data-set was used as an unlabeled set we can easily discern a block-structure where the images in the same class are similar to each but dissimilar to the images of the other class (images in the same class correspond to first 10 entries and second 10 entries respectively and brighter colors indicate higher similarity between the data-points). Note that the block structure is not evident when a dissimilar set of unlabeled training examples is used, i.e. the 'Odyssey' unlabeled data-set. Test performance and alignment values are also indicated

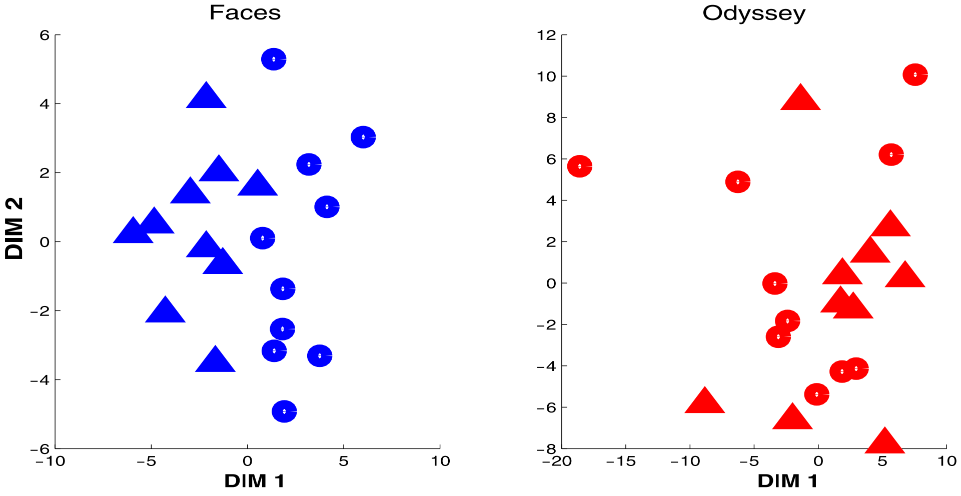

Fig. 11 Visualization of Fisher Scores extracted from models created using different sets of unlabeled training data. Each plot was the result of learning with a different set of unlabeled training data, namely 'Faces' (*Left*) and 'Odyssey' (*Right*). The circle and triangles are two different individuals for which Fisher Scores were extracted from the generative model. Note that none of these images were in the set of unlabeled training data used to create the 'Faces' generative model. Distances in feature space were computed as $d_{ij} = \|\boldsymbol{\phi}(x_i) - \boldsymbol{\phi}(x_j)\|$, where $\boldsymbol{\phi}$ is the normalized Fisher Score. These were input to the MDS procedure which embeds the data in a 2-D Euclidean space while minimizing distance distortion. One can clearly see that the model learned on the Faces data (left) results in an embedding which is linearly separable (even in 2 dimensions), while the embedding obtained from the Odyssey data-set (right) is not linearly separable (different classes are represented by circles and triangles). This indicates that using 'Faces' as an unlabeled training set will result in a more robust classifier

from all these models into a single Fisher Score by concatenating the Fisher Scores from each individual model. However, this is an unsatisfactory procedure because we do not know how to weight the different contributions, which may operate at different scales. Hence, a natural idea is to weight the Fisher Scores of each component model. Taking this one step further, one could decide to attach a different weight to each individual dimension of the entire feature vector, not just to each individual component model. This choice of parametrization is problem dependent: too many parameters

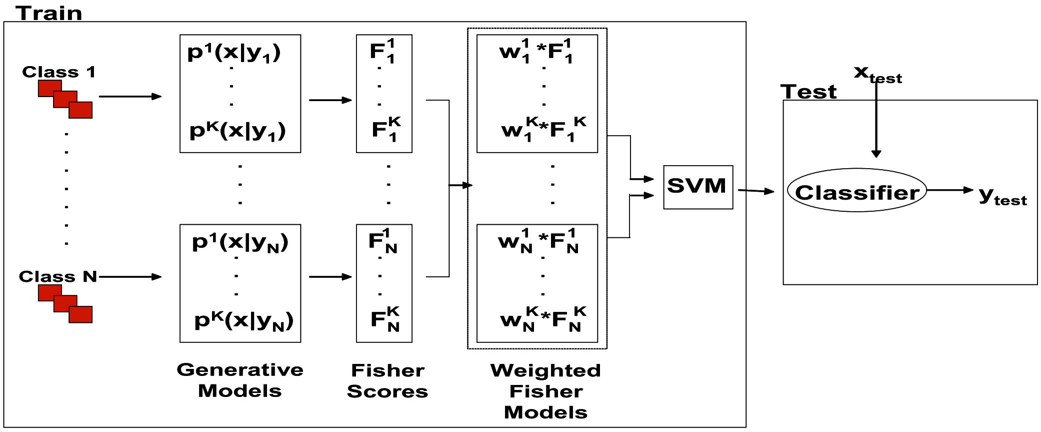

Fig. 12 Overview of proposed visual category recognition algorithm which automatically weights the importance of different models

may still lead to overfitting and may be too computationally expensive.

In this section we explore the potential of learning the weights for the various contributions to the kernel. This has lead to interesting insights; for instance that a certain type of detector is much better to build a face model than another type of detector. In some sense, by learning the optimal combination of kernels we are partially relieved from the task to pick the best components to model a certain object class. Instead we let the data decide which components are most important for the classification task. This information could subsequently be used to remove irrelevant components, thus improving the computational efficiency of the resulting classifier.

6.1 Leave-One-Out Span Bound

Our proposed method consists of three steps which are illustrated in Fig. 12: (1) Train an ensemble of generative models separately for each class, (2) extract Fisher Scores to construct Fisher Kernels for all the models separately, (3) train an SVM classifier and "learn the kernel" by weighting the different kernels. Finding the weights is achieved by minimizing a bound on the leave-one-out error (Chapelle et al. 2002). Next, we will provide details for these steps.

The general form of the kernel that we consider is,

$$K^{\mathrm{RBF}}(I_i, I_j) = \exp\left(-\frac{1}{2}\sum_{\mathcal{M}} w_{\mathcal{M}} \|\boldsymbol{\phi}_{\mathcal{M}}(I_i) - \boldsymbol{\phi}_{\mathcal{M}}(I_j)\|^2\right) \tag{12}$$

where we have introduced separate weights $w_{\mathcal{M}}$ for each model. To tune the weights $w_{\mathcal{M}}$ and the regularization constant C in the SVM, we adopt the method proposed in Chapelle et al. (2002). In this method a smoothed version of the "span-bound" of the leave-one-out (LOO) error[4] is minimized,

$$T(\mathbf{w}, C) = \frac{1}{N}\sum_{n=1}^{N} \sigma_T(\alpha_n S_n^2 - 1) \quad \text{with}$$

$$S_n^2 = \frac{1}{[(K_{SV} + \mathrm{Diag}(\eta/\boldsymbol{\alpha}))^{-1}]_{nn}} - \eta/\alpha_n$$

where $\sigma_T(x) = 1/(1 + e^{-x/H})$ is the sigmoid function, H is the temperature (set to $H = 100$), η is a smoothing parameter (set to $\eta = 0.1$) and K_{SV} the kernel evaluated at the support vectors. The parameters $\boldsymbol{\alpha}$ are the dual weights in a 2-norm soft margin SVM. The 2-norm SVM is convenient because the regularization parameter C can be considered as a parameter of the kernel function, alongside the weights $w_{\mathcal{M}}$ (hence the notation $\boldsymbol{\theta} = (\{w_{\mathcal{M}}\}, C)$ in the following). We invite the reader to explore Chapelle et al. (2002) for more details and a more complete explanation of this procedure.

The chief advantage of this smoothed span bound is that it can be efficiently minimized using gradient descent. The gradients can be written as,

$$\frac{\mathrm{d}T(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial T(\boldsymbol{\theta})}{\partial K_{SV}}\frac{\partial K_{SV}}{\partial \boldsymbol{\theta}} + \frac{\partial T(\boldsymbol{\theta})}{\partial \boldsymbol{\alpha}}\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{\theta}}. \tag{13}$$

While the first partial derivative is straightforward, the second requires some more thought because the dual weights $\boldsymbol{\alpha}$ are the solution of the 2-norm SVM. For more details on how to compute it we refer to Chapelle et al. (2002). Discontinuities in this derivative are expected when data-cases jump in or out of the support vector set. However, the gradient descent with line search procedure works well in practice. Note that for each gradient step we need to solve

[4] We show the case without bias term for simplicity, but bias was included in our experiments.

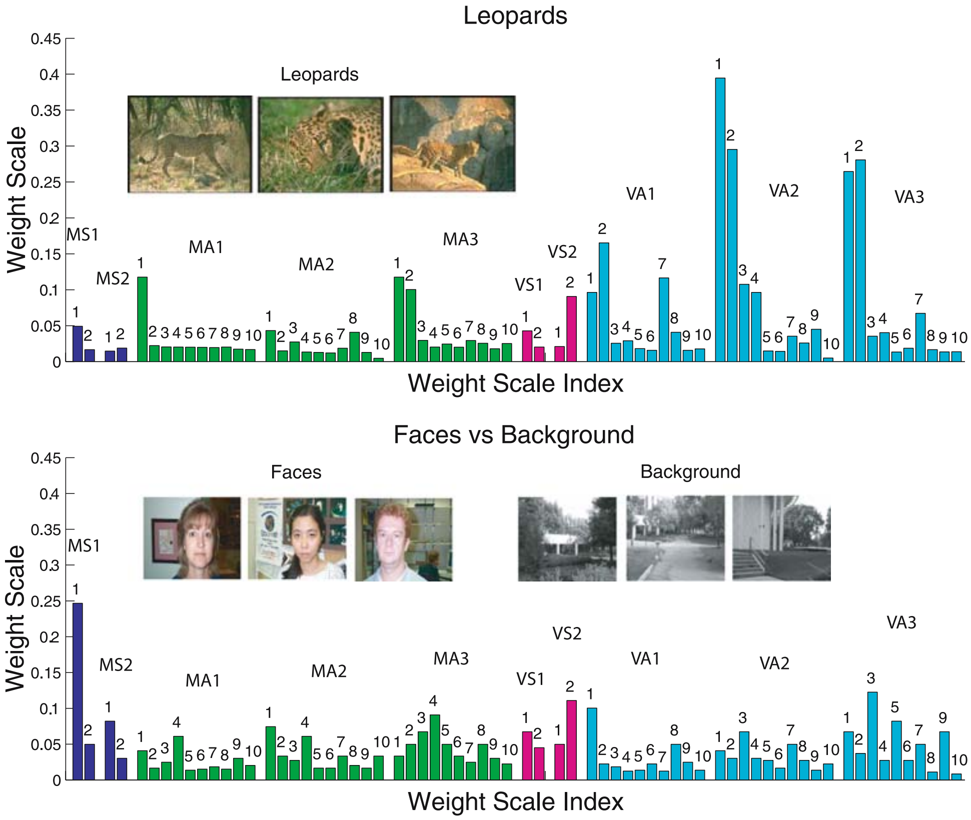

Fig. 13 Determining which model parameters are most useful for a particular classification task. Higher weights indicate more important features. Fisher Scores are extracted from a 3 part diagonal covariance model with 10 dimensional appearance model of the foreground class using the KB detector. 100 training examples were used. Position is conditioned on location of first model part so the shape model is only optimized over 2 parts. M/V indicates mean/variance, S/A indicates Shape/Appearance and 1/2/3 indicates the part within a model. Each group has 2 or 10 bars corresponding the number of dimensions. E.g. "VA2" indicates a group of 10 variance parameters for the appearance model of part 2. (*Top*) Leopards vs. Background classification task. Early coefficients in the appearance model are the most useful dimensions for classification. (*Bottom*) Faces vs. Background classification task. The Shape model appears to be more useful for classification. The consistent detection of facial features by the KB detector makes the shape model relatively more important

a QP for the SVM. However, this procedure is much more efficient than cross-validation which would need to check a number of grid-points exponential in the number of parameters, which is at least a dozen in our experiments.

7 Experiments with Combinations of Kernels

We tested our system which weights different models on numerous object categories. Each image within a category is associated with a class label, however the objects are not segmented within an image. We used some of the classes from previous experiments and collected an additional set of classes consisting of 200 images of 3 different faces taken against varying backgrounds and lighting conditions. Examples of the face images are shown in Fig. 1. For the 2-part models up to 100 detected interest points were used, and for the 4-part models up to 20 interest points. Typically 100 training images and up to 250 testing images were used.

Training a classifier proceeds as follows: First, learn a suite of generative models for each class using the training data only. In case we classify against background we only train models on the foreground class. Next, extract Fisher Scores from all models and train weights $w_{\mathcal{M}}$ by minimizing the span bound. The weighted scores are used to construct a single kernel K_{RBF} which is used to train a 2-norm SVM, still only using training data. The kernel and dual weights $\boldsymbol{\alpha}$ for the support vectors are then used in the usual way to predict the class label of test cases.

7.1 Feature Selection

The generative models contain numerous parameters from which Fisher Scores are extracted and it is unclear, in general, which components of the model(s) are important for classification tasks. Figure 13 illustrates the resulting weights after minimizing the LOO bound. We observe

Table 4 The effect of weighting and removing features on performance. Unweighted: performance using XVal on a single weight for all dimensions and a slack parameter. Weighted: performance after minimizing the LOO-span bound on the weights of each model. 10%/20%: performance using only the top 10% and 20% of the relevant dimensions. −10%: performance using the worst 10%. Others: the performance of previous constellation model algorithms. Note that Fergus et al. (2003) uses 6-part models including occlusion and typically 2–4 times more training examples

	Unweighted	Weighted	10%	20%	−10%	Others
leopards vs. BG	91.3	94.3	89	90	62	88 (Fergus 2005)
P1 vs. P3	86	89.8	82.2	81.1	67	–

Table 5 The effect of weighting and removing models on performance. Unweighted/Weighted/Others: same as Table 4. 1/2: the performance using only the best model/performance using the first and second best model. −1: performance using the worst model. 100 training images were used

	Unweighted	Weighted	1	2	−1	Others
motorcycles vs. BG	92.6	92.6	77.2	89.6	72	96.7 (Fergus 2005)
leopards vs. BG	93.4	94.8	90.5	93.7	69.1	88 (Fergus 2005)
airplanes vs. BG	89.8	91.4	80.6	87.2	65.4	93.3 (Fergus 2005)
P2 vs. P3	83.3	92.4	92.7	93.0	67.9	–

Table 6 Performance on discriminating between faces in the People Face data-set. KB/mHar/mHes/DoG: performance using Appearance/Shape models created using only the given detector. Unweighted: performance using models of all detectors with a single weight. Weighted: performance after LOO optimization of the weights for the models. Notice the relatively poor performance using only the KB or mHes on these classification tasks. 50 training images used, 2-part, 20 PCA coefficient models

	KB	mHar	mHes	DoG	Unweighted	Weighted
P1 vs P2	76.6	82.8	72.3	61	88.2	92.8
P1 vs P3	64.3	85.2	73.6	69.1	89.7	93.6
P2 vs P3	74.6	82.4	81.1	64.7	90.1	92.8

that the relative weight of features vary for each classification task, thereby giving us a deeper understanding of the importance of these model components for a particular task. Table 4 illustrates the effects on performance as subsets of 'good' and 'bad' features are removed. We observe that the weights obtained from the LOO procedure are good indicators of generalization ability of a particular feature. Parameter selection for the "unweighted" procedure was done using 10 fold cross validation (XVal) by varying both C and a single weight for all the models W in log steps from $[-9:3:9]$.

7.2 Model Selection

We have argued above that selecting for individual features becomes computationally difficult and subject to over-fitting when the number of LOO optimized parameters is too high. We address this issue by assuming each model has the same weight for each of its extracted fisher scores (i.e. the weights are tied across features within a model). We apply the LOO Span Bound described above to appropriately weight the models for maximum generalization performance. We generate separate models based on: (1) Shape and Appearance, (2) 2/4 model parts, (3) 4 different interest point detectors. This results in 16 models for each class. Figure 15 shows the evolution of the weights and the corresponding change in test error on a typical classification task.

Figure 14 illustrates that the LOO procedure selects different combinations of models depending on the particular classification task at hand. These optimized combinations of models yield an increase in classification performance over the XVal procedure on the unweighted models (see Table 5 and Table 6).

In addition, we study the effect of training on a subset of the models based on their optimized weights from the LOO procedure (see Fig. 15 Left, and Table 5). We notice that training the Xval procedure with only the best models results in good performance, while training using only

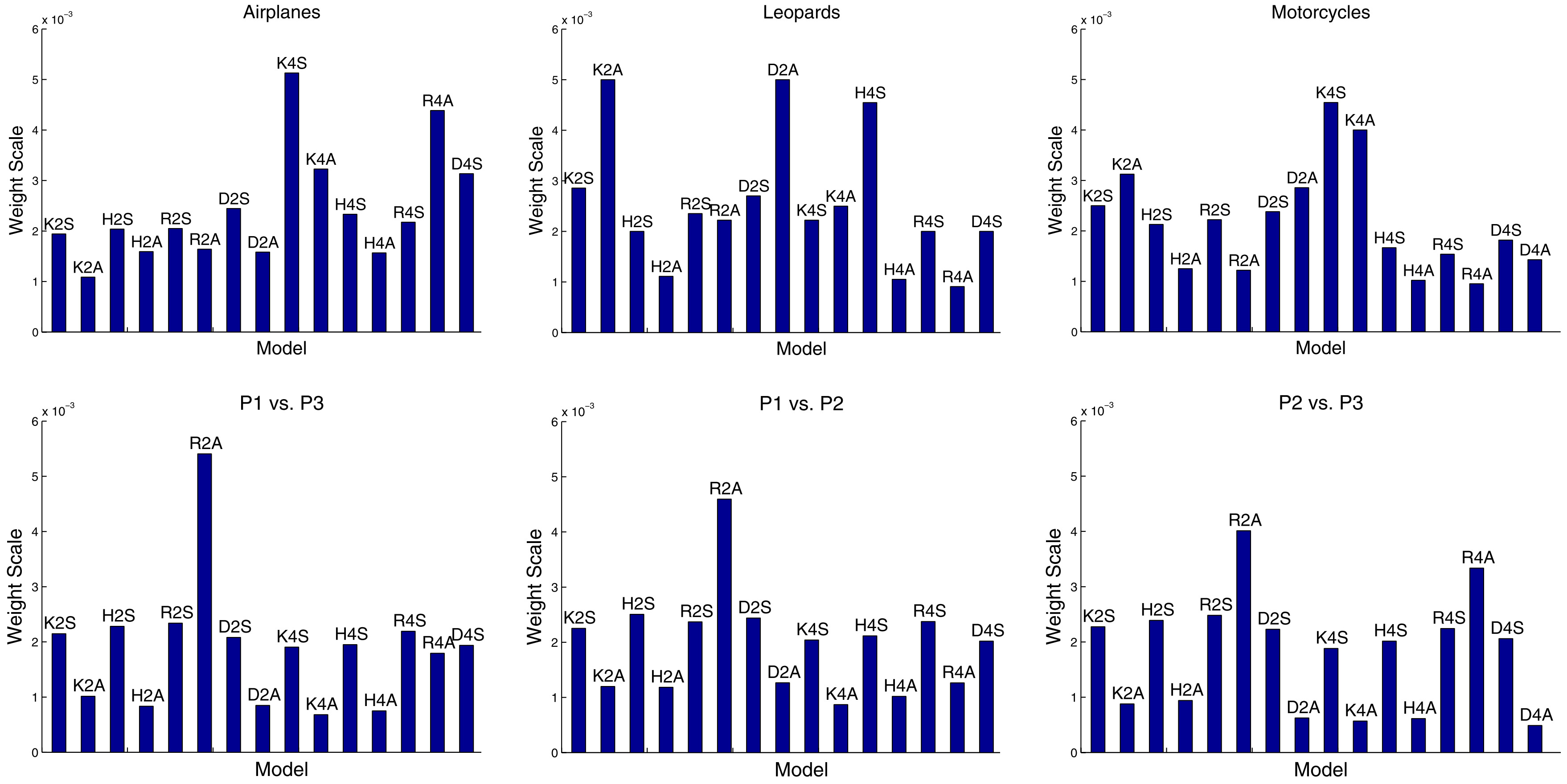

Fig. 14 The effects of weighting models on different classification tasks. K/H/R/D indicate KB/mHes/mHar/DoG detectors. 2/4 indicates Two/Four part models. S/A indicates Shape/Appearance models. E.g. "K2A" corresponds to a 2-part appearance model using the KB detector. Top Row. (*Left*) Airplanes vs. Background. (*Center*) Leopards vs. Background. (*Right*) Motorcycles vs. Background. Different combinations of models are selected for a particular classification task. *Bottom Row*. Performance on People Faces classification tasks. Notice that models created using mHar seem to be the best predictors of generalization ability. Performance using weighted models, from left to right, of 91.45, 92, and 92.42 respectively

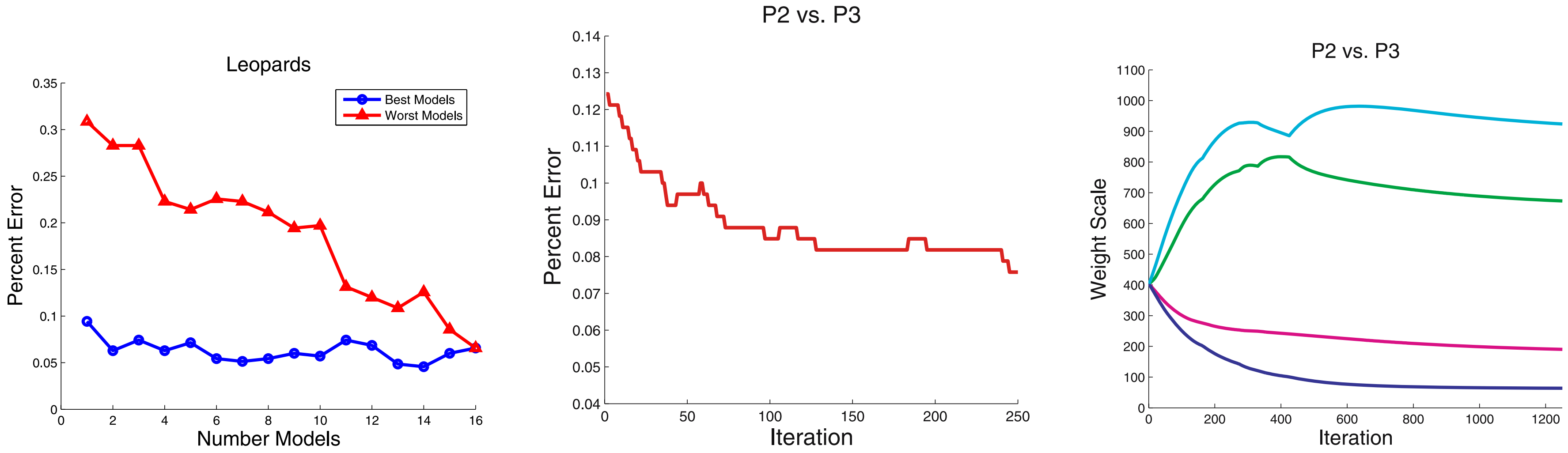

Fig. 15 (*Left*) Performance on the Leopards vs. Background classification task as a function of the number of models used. *Circles* indicate when the best models are used, i.e. the index ‘2’ would indicate that the two best models were selected. *Triangles* indicate that the worst models were used, i.e. index ‘2’ indicates that the worst two models were used. Models were weighted equally during classification. Each point generated by conducting Xval over the weights and slack parameter using only the specified models. (*Center*) Typical example of test error change during the LOO procedure using initialization parameters from Xval as a function of the number of iterations. (*Right*) Typical evolution of model weights for a combination of 4 models as a function of the number of iterations. Plot of weight scales where the weight scale is defined as function of the model weight w: $\sigma^2 = 1/w_{\mathcal{M}}$. Note that certain models become weighted more or less heavily emphasized as we progress in optimization

the models with poor weights results in low classification performance. This procedure can be used to select the best models for a particular classification task, allowing for computational savings during detection.

7.3 Integrating Unlabeled Data and Kernel-Combination—the Caltech 101

We next performed an experiment using our hybrid model which integrated both the use of unlabeled data in a semi-supervised learning paradigm as well as combining multiple models into a single classifier.

The Caltech 101 object category data-set[5] consists of 101 object categories with varying numbers of examples in each category (from about 30 to over 1000). The challenge of this data-set is to learn representations for many different object classes using a limited number of training examples. The variability of this data-set is mostly evident between different categories of objects rather than within a single category. Objects within a particular category are often somewhat homogeneous in both appearance and pose.

For our experiments we used the following experimental paradigm. First we created a broad underlying generative model using 5 randomly selected training images from all 101 classes. In the parlance of Sect. 5, this set of images corresponds to our set of unlabeled trained examples. Generative models from this set of unlabeled training examples were created using interest points detected from three different detectors, KB, mHess, mHar. Fisher Scores were then extracted from the generative models for all classes within the Caltech 101. Fisher Scores from each generative model were concatenated to form a single long vector to be used in the SVM classifier. To train our SVM classifier we used 15 training examples and up to 30 test examples. We used the same cross-validation procedure to find the hyper-parameters as described in the multi-class experiments above.

Figure 16 illustrates our results. These results show that utilizing the hybrid approach yields substantial improvements in classification performance over the generative approach and that additional performance gains are realized when generative models created using different underlying feature detectors are combined to form a single classifier. These experiments illustrate the utility in using a semi-supervised approach as well as combining kernels in improving classification performance.

8 Discussion and Conclusions

We have explored a method for obtaining discriminative classifiers from generative models of visual categories. It works in two steps: first train generative models, then from those generative models calculate Fisher Kernels and use them for classification. This method achieves the best of both the generative and discriminative worlds: for generative approaches it is robust to occlusion and clutter, it may be trained from few examples, it benefits from prior knowledge. Additionally, the generative part of the model may be trained incrementally. Its discriminative nature results in superior classification performance.

Our experiments in Sect. 4 show that the performance of our hybrid approach is not inferior to that of a traditional generative constellation model. Rather, performance is significantly better there and is in line with the current results in the literature. The advantage of the hybrid approach is particularly evident when the categories to be classified are very similar, such as the faces of different people. In addition, we controlled for possible overfitting to background statistics and found that this may be an issue.

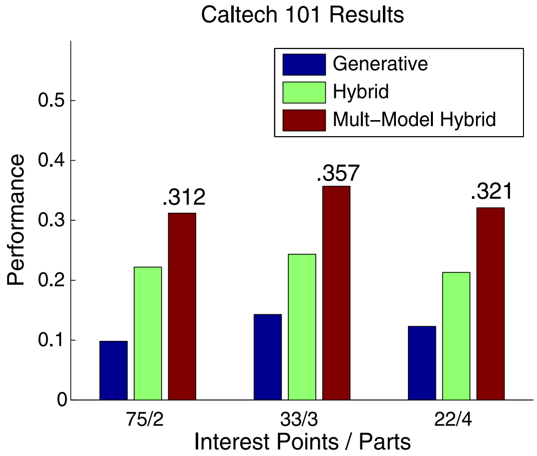

Fig. 16 Performance on the Caltech 101 object category data-set when different numbers of parts and interest points are used. Each set of columns indicate a different number of parts and interest points used, i.e. 33/3 indicates that a maximum of 33 interest points were detected in each image and that a 3 part model was used. Performance is measured by first calculating the percent of correctly classified points in each class and then taking the average over all classes (this corresponds to the average of the main diagonal of the confusion table). *First column in each set*: pure generative approach using interest points from the KB interest point detector. I.e. a generative model is created for all 101 categories and test examples are assigned to the generative model with the highest posterior probability. *Second column*: hybrid approach where Fisher Scores were extracted from a generative model made using the KB interest point detector. See text for more details. *Third column*: hybrid approach when three (KB, mHar, mHess) different interest point detectors are used and a generative model is created for each detector. Fisher Scores are concatenated into a single vector. Fei-Fei et al. (2004) managed 16% using a constellation model with 3 parts and integrating prior information into their model. For a similar 3-part constellation model with no prior information we achieve 14.3% using only the generative models, 26.1% using our hybrid approach, and 35.7% using our multiple model hybrid approach. 15 training examples were used for all experiments

[5] Available at http://www.vision.caltech.edu/html-files/archive.html.

Section 5 shows that our hybrid architecture lends itself readily to incorporating unlabeled examples. This is achieved by training generative models on large sets of unlabeled pictures which may contain relevant information. This process provides considerable performance improvement when learning specific categories (faces in our experiments) with very few training examples. As one would expect, we find that these results vary with the statistics of the images used to construct the generative model. Figure 9 shows that learning the statistics unrelated categories such as Text or Leopards will not help in distinguishing between Faces, while the most useful generative model has the same statistics as the Faces data-set.

In Sect. 6 we show that multiple models may be readily combined within our hybrid method. Our experiments in Sect. 7 suggest that the system is able to determine automatically which models provide the most valuable information for a given classification task. This indicates that a higher level of automation has been achieved: we do not need to ask an expert vision engineer to craft the best recognition strategy for a given task; rather, we can let our hybrid system self-tune to use whatever information is most valuable. Finally, we illustrate that combining both our semi-supervised learning approach and the ability to combine multiple models yields strong performance on the Caltech 101 object category data-set and that this performance is far superior to that of the corresponding generative approach.

References

Burl, M., & Perona, P. (1996). Recognition of planar object classes. In *Computer vision and pattern recognition (CVPR)* (p. 223).

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Crowley, J. L. (1984). A representation for shape based on peaks and ridges in the difference of low pass transform. In *Pattern recognition and machine intelligence (PAMI)*.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, *39*, 1–38.

Dorko, G., & Schmid, C. (2005). *Object class recognition using discriminative local features* (Technical Report RR-5497). INRIA.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples. In *Computer vision and pattern recognition (CVPR) workshop on GMBV*.

Fergus, R. (2005). *Visual object recognition*. Thesis, Department of Engineering Science, University of Oxford.

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *Computer vision and pattern recognition (CVPR)* (Vol. 2, p. 264).

Gold, C., Holub, A., & Sollich, P. (2005). Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. In *Neural Networks*.

Holub, A., Welling, M., & Perona, P. (2005). Combining generative models and fisher kernels for object class recognition. In *International conference on computer vision (ICCV)*.

Holub, A., & Perona, P. (2005). A discriminative framework for modeling object class. In *Computer vision and pattern recognition (CVPR)*.

Jaakkola, T., Diekhans, M., & Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems (NIPS)* (Vol. 11, pp. 487–493).

Jaakkola, T., & Haussler, D. (1999). Probabilistic kernel regression models. In *Proceedings of the seventh international workshop on artificial intelligence and statistics*.

Kadir, T., & Brady, M. (2001). Saliency, scale and image description. *International Journal of Computer Vision*, *45*(2), 83–105.

Leibe, B., & Schiele, B. (2004). Scale-invariant object categorization using a scale-adaptive mean-shift search. In *DAGM-symposium* (pp. 145–153).

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Ng, A., & Jordan, M. (2002). On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In *Advances in neural information processing systems (NIPS)* (Vol. 12).

Opelt, A., Fussenegger, M., Pinz, A., & Auer, P. (2004). Weak hypotheses and boosting for generic object detection and recognition. In *European conference on computer vision (ECCV)* (pp. 71–84).

Opper, M., & Winther, O. (2000). Gaussian processes and svm: Mean field and leave-one-out. In *Advances in large margin classifiers* (pp. 311–326). Cambridge: MIT Press.

Schneiderman, H. (2004). Learning a restricted Bayesian network for object detection. In *Computer vision and pattern recognition (CVPR)* (pp. 639–646).

Schoelkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge: MIT Press.

Seeger, M. (2002). Covariance kernels from Bayesian generative models. In *Advances in neural information processing systems (NIPS)* (Vol. 14, pp. 905–912).

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Torralba, A., Murphy, K. P., & Freeman, W. T. (2004). Sharing visual features for multiclass and multiview object detection. In *Computer vision and pattern recognition (CVPR)*.

Tsuda, K., Akaho, S., Kawanabe, M., & Müller, K.-R. (2003). *Asymptotic properties of the fisher kernel*. citeseer.ist.psu.edu/tsuda03asymptotic.html.

Ullman, S., Vidal-Naquet, M., & Sali, E. (2002). Visual features of intermediate complexity and their use in classification. In *Nature neuroscience* (pp. 682–687).

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley–Interscience.

Vasconcelos, N., Ho, P., & Moreno, P. (2004). The Kullback–Leibler kernel as a framework for discriminant and localized representations for visual recognition. In *European conference on computer vision (ECCV)* (pp. 430–441).

Wallraven, C., Caputo, B., & Graf, A. B. A. (2003). Recognition with local features: the kernel recipe. In *International conference on computer vision (ICCV)* (pp. 257–264).

Weber, M., Welling, M., & Perona, P. (2000). Towards automatic discovery of object categories. In *Computer vision and pattern recognition (CVPR)* (p. 2101).