Combining Classifiers for Improved Multilabel Image Classification

Martin Antenreiter, Ronald Ortner, and Peter Auer

University of Leoben, A-8700 Leoben, Austria
{mantenreit,rortner,auer}@unileoben.ac.at

Abstract. We propose a stacking-like method for multilabel image classification. Our approach combines the output of binary base learners, which use different features for image description, in a simple and straightforward way: The confidence values of the base learners are fed into a support vector machine (SVM) in order to improve prediction accuracy. Experiments on the datasets of the Pascal Visual Object Classes challenges (VOC) of 2006 and 2007 show that our method significantly improves over the performance of the base learners. Our approach also works better than more naive approaches for combining features or classifiers.

1 Introduction

The idea of combining classifiers in order to get improved prediction accuracy has been considered by many researchers (see e.g. [6, 17] for an overview and further references) and has sparked the development of seminal methods such as *boosting* [11, 26]. Here we consider *multilabel* learning problems where each instance may have several labels (unlike in *multiclass* problems where each instance is assigned to a unique class, i.e., has a single label). We present a very simple method for multilabel learning based on the combination of binary base classifiers. That is, we propose to train for each label a binary base classifier. Then we feed the output (i.e., the confidence values) of these binary base learners into a support vector machine (SVM) in order to improve prediction accuracy.

The basic concept underlying this simple approach resembles *stacking* [28] methods: In the stacking framework the output of several (distinct) base classifiers is combined by a meta-level learner to give an improvement in (binary or multiclass) classification. In the multiclass case usually multiclass classifiers are used as base learners and as meta-level learners (cf. [8]). While the idea behind stacking is that the metalearner shall be able to combine the base learners in a more sophisticated way than doing simple voting or cross-validation [28], in our method the meta-level learner shall grasp interdependencies between the single classes that the base learners have not properly captured.

There has been some discussion whether stacking really gives any improvement over choosing the best base classifier [8] (see also [25] for a related discussion). In any case, it is known that the success of stacking methods depends on

the choice of the meta-level learner as well as the kind of input this learner takes from the base learners [27]. Recent suggestions (see [8] for an overview) usually use the probability distributions predicted by the base learners (in some form) as input for the meta-level learner. Similarly, we use the base learners' confidence scores, which is usually simpler than working with probability distributions, as obtaining the latter requires additional optimization methods [23]. The choice for the meta-level learners in stacking approaches ranges from nearest neighbor [20] to tree methods [8]. SVMs have been used as metalearners as well [1, 7, 18].

We tested our algorithm on the popular image classification databases, provided for the VOC challenges in 2006 and 2007 [10, 9]. Instead of combining essentially different base learners, we kept the base learning algorithm fixed, while using different features for describing the image data. Results show that the combined classifiers outperform the base classifiers in every experiment, which indicates that the combined classifiers are indeed able to extract interdependencies between the individual classes and also the individual classifiers.

The rest of the paper is organized as follows. The following section describes our algorithm in more detail and discusses some related work. Section 3 describes the datasets and the experimental setup as well as the results. There, we also consider alternative methods for combining features or classifiers to which we compare our approach. The final section considers directions of future work.

2 Multilabel Classification

2.1 General Considerations

In multilabel classification problems there are usually interdependencies between classes. For example, when labeling images it is rather unlikely that an image containing a sheep also shows an aeroplane. On the other hand, images containing cars will usually also contain roads. The art of multilabel classification lies in reliably detecting and exploiting these interdependencies. A base classifier that is trained on enough examples will also learn these interdependencies. However, in the common case where the training set is not sufficiently large, a base learner for cars may not fully grasp the interdependency between cars and roads. This defect can be corrected by having a base learner for roads that will be able to learn roads from more training examples than just those where also cars appear. That way, combining the road classifier with the car classifier in a suitable way will also improve the performance for classification of cars.

Our algorithm uses binary base classifiers that are trained to recognize single classes. In order to better capture interdependencies between the individual classes we propose combining the confidence scores of the different base classifiers by simply feeding them into a support vector machine (SVM).

2.2 The Basic Algorithm

We consider the following *multilabel* problem: Given is a set of training examples $\{x_1, \ldots, x_n\} \subset X$ together with labels for N different classes $\{C_1, \ldots, C_N\}$

(where each $C_k \subseteq X$). That is, for each training instance x_i and each class C_k the respective label is

$$y_i^{(k)} := \begin{cases} +1 & \text{if } x_i \in C_k \\ -1 & \text{otherwise.} \end{cases}$$

As the problem is assumed to be *multilabel* but not necessarily *multiclass*, $y_i^{(k)} = 1$ not necessarily implies that $y_i^{(\ell)} = 0$ for $\ell \neq k$. Thus, the classes C_k in general will not partition the instance space X.

We first train for each class C_k a corresponding (binary) base learner $h^{(k)}$ that shall be able to predict the labels $y_i^{(k)}$ well. More generally, when using M distinct classification algorithms for each single class C_k, we have a total of NM base classifiers. Each base classifier returns a confidence score s for each training example x_i. In our case this will be a real value (the distance to the separating hyperplane) that is positive if the classifier predicts that x_i is in the target class and negative otherwise. However, more generally this score could also be a real number with a different interpretation (e.g. a probability distribution as in recent stacking approaches). Let $s_{jk}(x)$ be the confidence score returned by base classifier $h_j^{(k)}$ for class C_k on training instance x. The collected confidence scores are then combined by N metalearners, one for each class C_k, where the training set consists of the vectors

$$\mathbf{v}_i = \big(s_{1,1}(x_i), s_{1,2}(x_i), ..., s_{1,N}(x_i), s_{2,1}(x_i), ..., s_{2,N}(x_i), ..., s_{M,N}(x_i)\big) \qquad (1)$$

for all training instances x_i. The label of each $\mathbf{v}_i$ is simply the label $y_i^{(k)}$ of x_i with respect to class C_k.

2.3 Algorithm Specification for Image Classification

A state-of-the-art approach for image classification is the following: First, features are extracted from the images. These features then are clustered to generate a visual codebook. Based on that codebook a histogram of "visual words" for every image is built. This approach was inspired by the text classification community where it is called "bag-of-words". In text classification the input features are words, while in image classification descriptors take over this part. A powerful descriptor is the scale-invariant feature transform (SIFT) [19], although there are other very good descriptors for certain image classes. A common approach is to build for every descriptor type a histogram and concatenate these histograms.

Our base classifiers work with different features (as specified in Section 3 below) that are learned with a fixed learning algorithm (in our case either LP-Boost or Fisher kernels) to give the confidence scores. As metalearner SVMs [4] are deployed.

2.4 Some Related Work

There is a lot of work dealing with multilabel problems in far more complex ways than our straightforward approach. Thus, in [18] SVMs are used for combining different types of features for mapping of proteins to Gene Ontology.

Their method trains $\frac{N\cdot(N-1)}{2}$ binary classifiers for a problem with N labels. A function for combining and normalizing the output of the $\frac{N\cdot(N-1)}{2}$ binary classifiers is used and the normalization parameters have to be estimated. Additional knowledge of the problem domain is integrated by a directed acyclic graph to speed up the final classification.

Actually, it is quite common to model and use some additional context information in order to process the output of the base learners. This work is subsumed under the term of *Context Based Concept Fusion* (CBCF). For some references and an integrated approach see e.g. [24].

A simpler approach that has more in common with our method has been suggested in [13]. There it is proposed to use the base learners' output labels as additional coordinates in the training vectors. However, unlike our algorithm this does not consider the confidence of the base learners.

Compared to these exemplary alternative approaches, we find that our method is appealingly simple, and — as will be seen — works surprisingly well.

3 Experiments

3.1 Other Ways of Feature/Classifier Combination

In the experiments, we compared our approach with other ways to combine the base classifiers.

Using All the Features As a baseline on the first dataset (VOC 2006) we compare our approach to the more direct combination of the features by jointly using them for training the base classifiers. More precisely, the base learner — the boosting algorithm LPBoost — may choose in each boosting iteration the best single feature type for a decision stump. We used the Boosting approach with decision stumps, because it is very suitable for combining different kinds of feature types. (For further details see the experimental setup below.)

Binary Stacking In order to show that our algorithm profits from the confidence information of the other classes, we also did a comparison to the following alternative method where the metalearner uses for learning class C_k not the whole vectors $\mathbf{v}_i$ as given in (1) but only the confidence information for the class C_k at question. That is, the training vectors in this case are

$$\mathbf{v}_i = \big(s_{1,k}(x_i), ..., s_{M,k}(x_i)\big)$$

for each training instance x_i with label $y_i^{(k)}$. This corresponds to classical stacking on a binary classification problem, and we call this method *binary stacking* in what follows. Indeed, binary stacking with some minor modifications has been considered and empirically evaluated (among other multilabel algorithms) in [7].

The Best Binary Base Classifier Finally, we do a challenging comparison of our approach to the best binary base classifier. That is, we choose for the prediction of each class C_k the base classifier h_i that gives the best prediction accuracy on the test examples. Note that this classifier is usually unknown beforehand, so that choosing this best binary base classifier has an advantage over our method. However, even in this setting we show that our method gives better results.

3.2 Data Sets and Setup

We conducted experiments on the well-known image classification databases taken from the Pascal Visual Object Classes Challenges 2006 (VOC 2006) [10] and 2007 (VOC 2007) [9]. The VOC 2006 dataset contains 10 classes in 5,304 images, on which a total of 9,507 annotated objects can be found. The VOC 2007 dataset contains 20 classes in 9,963 images with 24,640 annotated objects. Both are multilabel datasets. These datasets are split into a fixed training, validation, and test set. In the VOC 2006 database the training set contains 1277 images. The validation set has 1341 images and the test set 2686 images. The VOC2007 database has 2501 training images, 2510 validation images, and 4952 test images.

The training of the base learners was done on the training dataset using the independent validation set for parameter selection. The selection of the kernel and the parameters was done using 5-fold cross-validation on the validation data. After that, we learned the combined classifiers using the validation data. All reported results are from the evaluation on the test data. As evaluation criterion we use the average precision as for the original VOC challenges.

3.3 Experiments on the VOC 2006 Dataset

In the experiments on the VOC 2006 dataset, we trained for each of the ten classes a classifier using LPBoost [3, 5] together with an image descriptor taken from a set of nine different feature types. The first feature type uses texture statistics of segment regions from a segmentation algorithm [12]. All other feature types are calculated from regions of interest obtained from a scale invariant Harris-Laplace detector [21]. On those regions subsampled grayvalues, basic moments, moment invariants [14], SIFT [19], and PCA-SIFT [16] descriptors are calculated. Further, we obtain three additional feature types by intensity normalization of the subsampled grayvalue, of the basic moments, and of the moment invariants descriptors. An overview over the used feature types is given in Tab. 1.

For our algorithm we used each feature type together with LPBoost to obtain for each class a total of nine binary classifiers as base learners. The output (the distance to the separating hyperplane) of these base learners (for all classes) was then fed into an SVM metalearner as described in Section 2.2, see (1). For completeness we tested our approach with different kernels. However, results show that the choice of the SVM kernel function is not critical.

h_1 ... texture statistics of segments
h_2 ... subsampled grayvalues
h_3 ... subsampled grayvalues (intensity normalized)
h_4 ... basic moments
h_5 ... basic moments (intensity normalized)
h_6 ... moment invariants
h_7 ... moment invariants (intensity normalized)
h_8 ... SIFTs
h_9 ... PCA-SIFTs

Table 1. Feature types for the experiments on the VOC 2006 dataset.

We compared our approach to the best of the base classifiers. An overview of the performance of each descriptor on the ten classes can be found in Tab. 2. It can be seen from Tab. 3 that our combined classifier outperforms the individual classifiers, even if we choose for each class that base classifier that gives the best performance on the test set.

As already indicated before, another comparison was made to the algorithm where the weak learner of LPBoost may choose in each boosting iteration one reference feature among the nine different feature types. Thus, in this setting the boosting algorithm is not restricted to a single feature type and may choose the best feature with the optimal threshold. This approach (denoted 'original classifier $h_{1..9}$' in Tab. 3 and 1) has been used in the VOC 2006 Challenge and is used as a base line for our experiments. For more details see [2]. While the results for this alternative algorithm sometimes improve even over the best single classifier, it is still outperformed by our algorithm. The collected results can be found in Tab. 3.

class	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9
bicycle	17.45	40.94	40.56	41.75	39.40	37.94	31.17	**56.84**	54.73
bus	9.57	25.94	**30.17**	25.38	19.39	13.60	16.30	16.03	25.62
car	40.78	**70.72**	69.99	48.74	51.33	38.17	41.41	56.90	60.47
cat	16.23	**29.60**	28.69	15.49	15.30	14.53	14.89	15.68	15.13
cow	8.99	18.15	15.40	17.62	18.74	10.18	17.48	17.64	**26.14**
dog	14.08	20.06	20.22	18.12	14.36	15.92	15.43	**22.86**	15.69
horse	9.94	12.29	16.86	**19.58**	13.08	14.65	11.49	10.06	12.24
motorbike	13.22	27.10	29.11	32.96	**39.90**	28.35	26.19	10.54	12.26
person	29.66	39.92	36.94	38.34	**43.47**	36.01	42.55	31.51	31.97
sheep	8.99	24.99	25.34	21.85	19.47	11.06	15.92	29.42	**36.37**
avg	16.89	30.97	31.33	27.98	27.44	22.04	23.28	26.75	29.06

Table 2. Average precision of the individual classifiers on the VOC 2006 dataset in percent. Bold values indicate the best classifier on a given class for the test set.

class	$\max(h_1, \ldots, h_9)$	original $h_{1..9}$	our (linear)	our (polynomial)	our (RBF)
bicycle	56.84	61.12	**61.36**	56.16	60.77
bus	30.17	27.32	**51.66**	50.00	51.54
car	70.72	70.92	74.03	**76.30**	74.83
cat	29.60	24.41	37.13	**41.15**	37.98
cow	**26.14**	18.92	23.41	24.20	25.56
dog	22.86	25.88	32.16	34.95	**36.41**
horse	19.58	12.12	22.44	**27.44**	20.86
motorbike	39.90	33.19	47.09	46.92	**48.35**
person	43.47	35.16	42.55	**46.56**	43.04
sheep	36.37	29.39	**41.87**	37.07	40.55
avg	37.57	33.84	43.37	**44.07**	43.99

Table 3. Comparison of the best individual classifier, the classifier using all descriptor types in the beginning, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. Bold values indicate the optimal method.

Fig. 1 shows a comparison of our method with the binary stacking approach. For binary stacking we used the same nine base classifiers that are combined by an SVM. We report only the results of binary stacking with an RBF-kernel SVM as metalearner, which performed best. The resulting performance of binary stacking is on some classes slightly better than our base line where all features are used from the beginning. However, on some other classes binary stacking suffers a performance decrease of up to 9.6%. The mean average precision of binary stacking is 30.52%, which compared to our base line means a performance loss of 3.32%. These experiments may also confirm doubts concerning the utility of stacking. However, as already mentioned before, usually class probabilities instead of confidence values are used for stacking. The choice of the latter may affect the results to the negative. Indeed, a similar stacking method [1] in an image classification problem where class probabilities are the input for the metalearner has been more successful. That confidence values work fine in our proposed method can also be interpreted the way that our metalearner SVMs do the normalization that also has to be done when trying to obtain probability distributions from confidence values.

3.4 Experiments on the VOC 2007 Dataset

In the experiments on the VOC 2007 database we had only access to two base classifiers. The first classifier h_1 is based on texture information using the SIFT descriptor [19], while the second classifier h_2 is based on Gaussian weighted local color information (h_2). Both descriptors are extracted from a dense grid at five different scales. Each classifier is learned with the Fisher kernels framework [22], which gives state-of-the-art performance on the VOC 2007 database. Tab. 4 shows that the performance of the classifier using the SIFT descriptor yields con-

class	$h_{1..9}$	stacking	our (RBF)
bicycle	**61.12**	56.77	60.77
bus	27.32	27.52	**51.54**
car	70.92	65.86	**74.83**
cat	24.41	14.75	**37.98**
cow	18.92	11.83	**25.56**
dog	25.88	17.55	**36.41**
horse	12.12	12.42	**20.86**
motorbike	33.19	30.21	**48.35**
person	35.16	34.25	**43.04**
sheep	29.39	34.00	**40.55**
avg	33.84	30.52	**43.99**

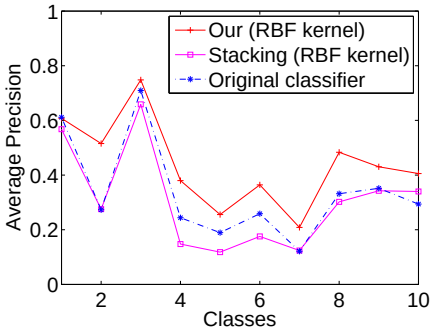

Fig. 1. Comparison of the classifier using all descriptor types in the beginning, binary stacking, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. For binary stacking and our method we report the values obtained for the RBF kernel.

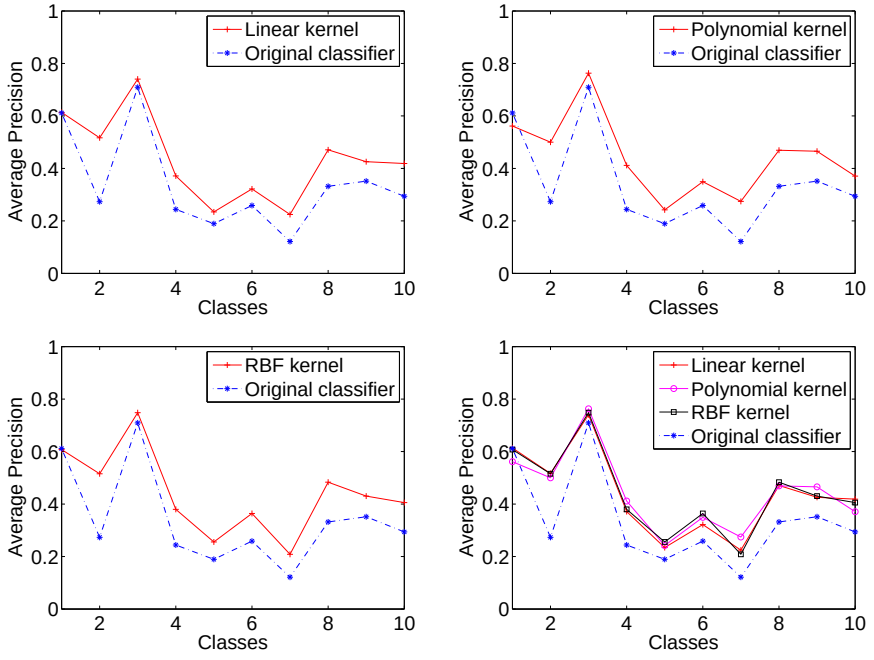

Fig. 2. Comparison of the classifier using all descriptor types in the beginning and our approach using different kernels on the VOC 2006 dataset. Results are in percentage using the average precision measure. The choice of the SVM kernel function can be seen to be not critical.

class	h_1	h_2	our (linear)	impr.	our (poly.)	impr.	our (RBF)	impr.
aeroplane	66.41	59.51	65.88	-0.53	65.47	-0.94	**66.71***	0.30
bicycle	47.31	35.45	51.09	3.78	53.23	5.92	**53.42***	6.11
bird	44.45	42.67	49.99	5.54	53.06	8.61	**53.47***	9.02
boat	58.87	41.12	63.21	4.34	**63.26**	4.39	62.38*	3.51
bottle	24.18	15.16	25.97	1.79	**27.53**	3.35	23.93*	-0.25
bus	**52.42**	34.24	51.65	-0.77	43.64	-8.78	45.75*	-6.67
car	70.70	56.47	70.67	-0.03	**73.32***	2.62	**73.32**	2.62
cat	45.30	39.49	44.87	-0.43	44.78	-0.52	**46.30***	1.00
chair	47.11	37.78	50.68	3.57	49.82	2.71	**50.72***	3.61
cow	31.25	15.03	29.00	-2.25	31.28	0.03	**32.99***	1.74
diningtable	38.21	35.75	42.29	4.08	43.12	4.91	**44.71***	6.50
dog	40.98	33.44	38.77	-2.21	40.42	-0.56	**41.95***	0.97
horse	67.77	64.48	71.44	3.67	73.46	5.69	**73.48***	5.71
motorbike	52.37	46.02	55.07	2.70	56.05	3.68	**57.85***	5.48
person	80.17	78.33	82.43	2.26	83.01	2.84	**83.22***	3.05
pottedplant	24.30	27.14	28.71	1.57	29.11	1.97	**32.92***	5.78
sheep	27.32	25.48	36.47	9.15	28.76	1.44	**38.79***	11.47
sofa	**44.36**	31.57	41.81	-2.55	40.67	-3.69	40.73*	-3.63
train	65.21	54.42	66.88	1.67	69.52	4.31	**69.87***	4.66
tvmonitor	41.34	34.26	44.54	3.20	**48.27**	6.93	46.66*	5.32
avg	48.50	40.39	50.57	2.07	50.89	2.39	**51.96**	3.46

Table 4. Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for our method with linear, polynomial and RBF kernel. The best method for each class is indicated by a bold entry. Starred values indicate which kernel is chosen by cross-validation.

sistently better results than the descriptor based on the local color information (with the only exception being the class 'pottedplant'). In spite of this and the fact that the information of two classifiers is quite limited, our method was able to improve the mean of the average precision across all the 20 categories by up to 3.46% (for the RBF kernel). When using cross-validation to choose the kernel, the RBF kernel is selected for all classes except one (cf. Tab. 4) giving the same average precision as for the RBF kernel. The collected results can be found in Tab. 4.

In Tab. 5 we compare our method with binary stacking. In this experiment a linear kernel gave the best results for the binary stacking, but it can be seen that our method gives better results for 14 classes. Our average improvement to the base classifiers is 3.46%, whereas binary stacking only improves by 1.66%.

4 Conclusion

While our presented method of combining classifiers for multilabel classification is appealingly simple, it works quite well. Our main goal was to investigate the

class	h_1	h_2	stacking	impr.	our (RBF)	impr.
aeroplane	66.41	59.51	**68.16**	1.75	66.71	0.30
bicycle	47.31	35.45	48.61	1.30	**53.42**	6.11
bird	44.45	42.67	49.37	4.92	**53.47**	9.02
boat	58.87	41.12	60.65	1.78	**62.38**	3.51
bottle	24.18	15.16	**25.76**	1.58	23.93	-0.25
bus	**52.42**	34.24	51.15	-1.27	45.75	-6.67
car	70.70	56.47	69.72	-0.98	**73.32**	2.62
cat	45.30	39.49	**46.86**	1.56	46.30	1.00
chair	47.11	37.78	45.18	-1.93	**50.72**	3.61
cow	31.25	15.03	30.96	-0.29	**32.99**	1.74
diningtable	38.21	35.75	38.51	0.30	**44.71**	6.50
dog	40.98	33.44	**43.20**	2.22	41.95	0.97
horse	67.77	64.48	70.61	2.84	**73.48**	5.71
motorbike	52.37	46.02	56.45	4.08	**57.85**	5.48
person	80.17	78.33	81.83	1.66	**83.22**	3.05
pottedplant	24.30	27.14	29.50	2.36	**32.92**	5.78
sheep	27.32	25.48	30.97	3.65	**38.79**	11.47
sofa	44.36	31.57	**45.15**	0.79	40.73	-3.63
train	65.21	54.42	67.76	2.55	**69.87**	4.66
tvmonitor	41.34	34.26	42.75	1.41	**46.66**	5.32
avg	48.50	40.39	50.16	1.66	**51.96**	3.46

Table 5. Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for binary stacking and our method with RBF kernel.

gain of our approach with respect to the base learners and simpler methods for combining features or classifiers. However, as higher complexity not necessarily results in improved performance [15, 8], it would of course be interesting to compare our elementary method to more complex algorithms for combination of classifiers. Another direction of future work is to test our approach on similar problems with different features available, such as in text classification.

Acknowledgments The authors would like to thank Teo de Campos from XEROX for providing the confidence scores for the experiments on the VOC 2007 dataset. This work was supported in part by the Austrian Science Fund FWF (S9104-N13 SP4). The research leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements n° 216886 (PASCAL2 Network of Excellence), and n° 216529 (PinView). This publication only reflects the authors' views.

References

1. Azizi Abdullah, Remco Veltkamp, and Marco Wiering. Spatial pyramids and two-layer stacking SVM classifiers for image categorization: A comparative study. In *International Joint Conference on Neural Networks (IJCNN 2009)*.

2. Martin Antenreiter, Christian Savu-Krohn, and Peter Auer. Visual classification of images by learning geometric appearances through boosting. In *Artificial Neural Networks in Pattern Recognition (ANNPR 2006)*, pages 233–243, 2006.
3. Kristin P. Bennett, Ayhan Demiriz, and John Shawe-Taylor. A column generation algorithm for boosting. In *Proceedings of the 17th International Conference on Machine Learning (COLT 2000)*, pages 65–72. Morgan Kaufmann, 2000.
4. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
5. Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.
6. Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
7. Anastasios Dimou, Grigorios Tsoumakas, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. An empirical study of multi-label learning methods for video annotation. In *7th International Workshop on Content-Based Multimedia Indexing (CBMI 2009)*, 2009.
8. Sašo Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, 2004.
9. Mark Everingham, Luc J.Ṽan Gool, Christopher K.Ĩ. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
10. Mark Everingham, Andrew Zisserman, Christopher K.Ĩ. Williams, and Luc J.Ṽan Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.
11. Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory (COLT 1990)*, pages 202–216, 1990.
12. Michael Fussenegger, Andreas Opelt, Axel Pinz, and Peter Auer. Object recognition using segmentation for feature detection. In *International Conference on Pattern Recognition (ICPR) (3)*, pages 41–44, 2004.
13. Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference (PAKDD 2004)*, pages 22–30. Springer, 2004.
14. Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/photometric invariants for planar intensity patterns. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision – Volume I*, pages 642–651. Springer, 1996.
15. Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.
16. Yan Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages 506–513. IEEE Computer Society, 2004.
17. Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.
18. Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, and Matthias Schubert. Using support vector machines for classifying large sets of multi-represented objects. In *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM 2004)*. SIAM, 2004.

19. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
20. Christopher J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58, 1999.
21. Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision (ICCV 2001)*, pages 525–531, 2001.
22. Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition (CVPR 2007)*. IEEE Computer Society, 2007.
23. John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Boston, 1999.
24. Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th International Conference on Multimedia 2007 (MM 2007)*, pages 17–26. ACM, 2007.
25. Ryan M. Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
26. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
27. Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
28. David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.