StyP-Boost: A Bilinear Boosting Algorithm for Learning Style-Parameterized Classifiers

Jonathan Warrell[1]
jwarrell@brookes.ac.uk

Simon Prince[2]
s.prince@ucl.ac.uk

Philip Torr[1]
philiptorr@brookes.ac.uk

[1] Oxford Brookes University
Oxford, UK
http://cms.brookes.ac.uk/research/visiongroup

[2] University College London
London, UK

Abstract

We introduce a novel bilinear boosting algorithm, which extends the multi-class boosting framework of JointBoost to optimize a bilinear objective function. This allows style parameters to be introduced to aid classification, where *style* is any factor which the classes vary with systematically, modeled by a vector quantity. The algorithm allows learning to take place across different styles. We apply this *Style Parameterized Boosting* framework (StyP-Boost) to two object class segmentation tasks: road surface segmentation and general scene parsing. In the former the style parameters represent global surface appearance, and in the latter the probability of belonging to a scene-class. We show how our framework improves on 1) learning without style, and 2) learning independent classifiers within each style. Further, we achieve state-of-the-art results on the Corel database for scene parsing.

1 Introduction

Many problems in computer vision benefit from modeling the interaction between two or more underlying variables, such as lighting and shape, or pose and appearance. In such cases, bilinear models have proved an effective means of representing the interaction. Bilinear models can be learned by classical techniques such as SVD, in generative frameworks using Expectation Maximization [12], or discriminatively using SVMs [3]. In this paper we present a bilinear boosting algorithm, *Style Parameterized Boosting* (StyP-Boost), which extends the multi-class boosting framework of JointBoost [14] to optimize a bilinear objective function. Our main application is to provide improved unary potentials for *object class segmentation*, the task of providing an object label for every pixel in the image. We emphasize though the generality of the framework, which could equally apply to object and boundary detection for example, as well as different boosting frameworks.

Recent approaches to object class labeling have formulated the problem in a Conditional Random Field (CRF) framework [5, 6, 7, 10, 11], which has the advantage of allowing multiple sources of information to be combined in a single discriminative model. An important component of these models are the *unary potentials*, which give a pixel-by-pixel estimate of the probabilities for each object class present. Typically, these are the output of a probabilistic classifier. Several methods have found the feature selection properties of boosting

to be valuable for such classifiers [7, 10, 11], giving state-of-the-art performance on several datasets when combined with powerful graph-cut algorithms for CRF inference.

Global sources of information have been shown to be important both in segmentation and other related problems. For instance, knowing the type of image (a forest scene, an indoor scene) can help us substantially in segmenting it. The utility of global context has been extensively demonstrated for the task of object detection in the work of Torralba *et al.* on 'contextual priming' [13, 14]. Here, global appearance features of a scene (GIST) can induce a prior weighting over an image, either directly [13] or after estimating scene-class [15], which is then used to weight detector responses for an object. Within object class segmentation approaches, there have been various methods for integrating image-level information into the task. Shotton *et al.* [9] for example investigate using an *image level prior*. This uses global features to predict the probability that each object class is present in the image as a whole. These probabilities are then used to weight the local probabilities, thus reducing errors caused by insufficient local information. An alternative approach is adopted in the multiscale CRF model of He *et al.* [5]. Here, a set of *global label features* is learned, each of which includes a distribution over labels at a set of coarsely subsampled locations across an image. Different global features can thus represent different scene classes via their label distributions, which are used to weight the unary estimates via the CRF framework (sampling is used to integrate across global features at test time).

A problem with both the image level prior and global label features is that the global information can only re-scale the unary responses by a prior probability of an object's presence, but cannot reinterpret local data for better discrimination. For instance, we may know in one environment that chairs are red and carpet is green, and in another vice versa, but without access to global knowledge about the scene the unary classifiers must respond equally to both objects for red and green inputs. An alternative model, the mixture of CRFs in He *et al.* [6] avoids this problem by learning a complete CRF for each scene-class, represented by a single mixture component. A gating function is used to weight the components based on global image features. Because separate unary potentials are learnt for each component, objects can be discriminated differently in each. However, a disadvantage of this approach is that we cannot share information between components during training: Since the unary classifiers are separate, we cannot use our knowledge of what a chair looks like in one environment to recognize it in another.

Bilinear models allow the interaction between two sets of variables to be modeled, and thus offer a solution to these kinds of problems. In [12], bilinear models were introduced in a generative context to model the interaction of style and content in a number of domains, including handwritten digit and facial imagery. More recently, a number of approaches have used bilinear models in a discriminative setting [3, 8], developing bilinear SVM approaches for tasks such as object detection and action recognition. A further application of bilinear models then, motivated by the concerns above, is to build classifiers for object class segmentation unary potentials which adapt with global image features. In the terminology of Tenenbaum *et al.* [12], we wish to introduce global *style parameters* into the classifiers, which could represent the probability of being in a certain scene-class for instance (e.g. urban, countryside), or position in a global appearance space (e.g. overall lighting, textural layout). While we could apply a bilinear SVM directly to this problem, motivated by the success of TextonBoost and related methods to object class segmentation [7, 10, 11], we develop a bilinear boosting algorithm, *Style Parameterized Boosting* (StyP-Boost), which responds to the problems above by 1) allowing the classifiers to adapt and interpret local data differently in response to different style parameters, and 2) also allowing them to exploit commonalities

to learn about objects across styles, and so make best use of the data at training time.

Section 2 summarizes the previous approaches to boosting on which ours is based, and Section 3 introduces our new StyP-Boost algorithm. We show the potential of our approach on two contrasting object class labeling tasks in Sections 4 and 5 (road surface parsing and scene parsing respectively), and finally discuss limitations and future directions in Section 6.

2 Previous Boosting Approaches

We briefly review the previous approaches to boosting on which we build, including AdaBoost.MH, JointBoost and TextonBoost. Traditionally, two-class AdaBoost has been derived from the exponential loss function, $J = \sum_i \exp(-z_i H(i))$, where i runs across the training instances, $z_i \in \{1, -1\}$ indicates if the example is in the positive or negative class, and $H(i)$ is the *strong learner* to be learnt. $H(i)$ is composed of *weak learners* $h(i)$, $H(i) = \sum_m h_m(i)$, which are selected successively over m rounds of boosting based on how well they minimize J. If the h's output is $\{1, -1\}$ we have DiscreteBoost, if they output real numbers RealBoost, and if we make only a Newton step at each round (i.e. do not find the best h) we have GentleBoost (see [4]). The quantity J itself is related to the margin [14], while a further variant, LogitBoost, substitutes the negative binomial log-likelihood for J.

A number of formulations for multi-class boosting have been proposed. In AdaBoost.MH [4], the exponential loss above is expanded in the following way to $k = 1...K$ classes:

$$J = \sum_{k,i} \exp(-z_{k,i} H(k,i)) \tag{1}$$

$$H(k,i) = \sum_m h_m(k,i) \tag{2}$$

where, $z_{k,i}$ is 1 if the instance is in class k and -1 otherwise. JointBoost [14], the algorithm we will go on to discuss, is an algorithm for minimizing (1) with a particular set of weak-learners. In Section 3 we will follow this formulation, but substitute a bilinear form of $H(k,i)$ into (2). We note however that the framework developed is not particular to JointBoost, and any of the boosting variants discussed above could be adapted to bilinear form. Particularly, the LogitBoost cost in many ways provides a more natural multi-class generalization than (1) via the multinomial likelihood, and could form an alternative starting point.

JointBoost proceeds at each round of boosting by optimizing a local cost function based on a second-order Taylor expansion of (1) about the weak learner to be added, h_m: $J_m = \sum_{k,i} w_{k,i}^m (z_{k,i} - h_m(k,i))^2$. Here, the weights w are recursively defined, and updated at each round via:

$$w_{k,i}^m = w_{k,i}^{m-1} \exp(-z_{k,i} h_m(k,i)) \tag{3}$$

The weak classifiers themselves are of the form:

$$h(k,i) = \begin{cases} a[v_n(i) > \theta] + b & \text{if } k \in S \\ \kappa^k & \text{otherwise} \end{cases} \tag{4}$$

where $v_n(i)$ returns the n'th feature for training instance i (assuming we have an enumeration of features), θ is a threshold, and a and b are constants. S is a subset of $\{1...K\}$, indicating which classes are to share the feature, while constants κ^k are assigned to the other classes to cope with imbalances of training data. An exhaustive search over weak-learners to maximize

J_m is typically infeasible. Given v_n, θ and S though, the other parameters can be found in closed form:

$$b = \frac{\sum_{k \in S,i} w_{ki} z_{ki} [v_{[r,t]}(i) \le \theta]}{\sum_{k \in S,i} z_{ki} [v_{[r,t]}(i) \le \theta]}, \quad a + b = \frac{\sum_{k \in S,i} w_{ki} z_{ki} [v_{[r,t]}(i) > \theta]}{\sum_{k \in S,i} z_{ki} [v_{[r,t]}(i) > \theta]}, \quad \kappa^k = \frac{\sum_i w_{ki} z_{ki}}{\sum_i w_{ki}} \quad k \notin S \tag{5}$$

JointBoost was applied to object class segmentation by Shotton *et al.* in the form of *TextonBoost* [11]. This follows the same form as above, but uses a specific set of features in (4), $v_{[r,t]}(i)$, which return the response of texton t within rectangular region r in relation to pixel i. Such features allow the classifier to make use of context as appropriate, by selecting rectangles over a large surrounding region. TextonBoost also adopts a particular search strategy to optimize $v_{[r,t]}$, θ and S, which cannot be found analytically, at each round of boosting:

Random feature selection: At each round of boosting, a set number (100) of features $v_{[r,t]}(i)$ are randomly selected for testing.

Logarithmic search over thresholds: A fixed set of θ's are tested, arranged equidistant on a logarithmic scale centered at zero.

Greedy search through object powerset for S: To avoid searching exhaustively through K^K for the set of shared object classes, each round begins by setting S to the null set, and greedily adds the class k which improves J_m most until no further improvement is possible.

The TextonBoost framework was extended recently to include quantized HOG and Colour-HOG visual words as well as textons in [7], and we shall refer to this extension as *Dense-Boost* following correspondence with the authors. This extension, combined with various CRF models, has achieved state-of-the-art results on several datasets, including MSRC and CamVid (see [7, 11]).

3 Style Parameterized Boosting (StyP-Boost)

We now describe a bilinear boosting algorithm based on the JointBoost/TextonBoost framework outlined above. We begin by noting that JointBoost itself can be considered to fit a sparse linear model in the space of all possible features, $h(i)$. For instance, if we enumerate all possible weak learners (e.g. we fix $a = 1$, discretize possible values for b/a and κ/a, let S run through the powerset K^K, and run through all $v_n(i)$ $n = 1...N$) then we can define a vector $\mathbf{y}_{i,k}$ containing an entry for every possible weak learner evaluated at i and k (i.e. $h(k,i)$). JointBoost can be seen to learn a sparse model, $H(k,i) = \mathbf{a}^T \cdot \mathbf{y}_{i,k}$, on these vectors.

We now consider introducing style parameter vectors $\mathbf{p}_i$ associated with each training instance. In the models we consider, these will be of length C, where C is the number of 'style categories' in our model, and the vectors are normalized probability distributions, where entry $p_{c,i}$ represents the probability instance i belongs to style category c. However, we note that that this restriction is not necessary, and we could treat the vectors as simply positions in a C-dimensional style-space. Using the same notation as above then, *StyP-Boost* seeks to learn a bilinear model of the form $H(k,i) = \mathbf{p}_i^T \mathbf{A} \, \mathbf{y}_{i,k}$. We will assume that the $\mathbf{p}_i$'s are fixed before training, and hence this is equivalent to a linear model in the Kröneker product of $\mathbf{p}_i$ and $\mathbf{y}_{i,k}$. A future direction is to allow the $\mathbf{p}_i$'s themselves to vary and find the best 'discriminative styles' during training (see Section 6).

Expressing the bilinear form just described in additive form, we rewrite (1) and (2) as:

$$J = \sum_{k,i} \exp(-z_{k,i} H(k,i)) \qquad (6)$$

$$H(k,i) = \sum_m [h_m(k,i) \cdot \sum_{c \in T_m} p_{c,i}] \qquad (7)$$

We note that the overall objective (6) remains unchanged. However, in (7) we now have an additional factor associated with each learner: $\sum_{c \in T_m} p_{c,i}$. Here, $T_m \subset \{1...C\}$ is the set of styles shared by weak learner m. Assuming the $\mathbf{p}_i$'s to be probabilities, we are thus weighting the weak learner by the combined probability the instance is in any of the shared styles. We can thus consider T_m to be an additional parameter which needs to be set at each round of boosting, and write $h'_m(k,i,T_m) = h_m(k,i) \cdot \sum_{c \in T_m} p_{c,i}$ to denote an 'expanded weak learner' which incorporates this extra parameter.

As an intuition, consider we are to recognize 'grass' at several different times of day and night (the 'style categories'). We might expect that weak learners based on the feature green should be shared by the day styles but not the night styles. Texture-based learners on the other hand might best be shared by all style categories.

Substituting (7) for (2), we require only minimal changes to the JointBoost training algorithm described in Section (2). Specifically, in the weight update (3) we simply substitute $h'_m(k,i,T_m)$ for $h_m(k,i)$. The analytic updates of (5) must be changed to incorporate the $\mathbf{p}_i$'s:

$$b = \frac{\sum_{k \in S,i} w_{ki} z_{ki} p'_i [v_{[r,t]}(i) \le \theta]}{\sum_{k \in S,i} z_{ki} (p'_i)^2 [v_{[r,t]}(i) \le \theta]}, \quad a+b = \frac{\sum_{k \in S,i} w_{ki} z_{ki} p'_i [v_{[r,t]}(i) > \theta]}{\sum_{k \in S,i} z_{ki} (p'_i)^2 [v_{[r,t]}(i) > \theta]},$$

$$\kappa^k = \frac{\sum_i w_{ki} z_{ki} p'_i}{\sum_i w_{ki} (p'_i)^2} \quad k \notin S \qquad (8)$$

where $p'_i = \sum_{c \in T} p_{c,i}$. We adopt the TextonBoost search strategies outlined above to find $v_{[r,t]}, \theta$ and S for each weak learner. However, we now also need to search for T, the set of shared styles for the new weak learner. In contrast to the greedy additive search through the class powerset for S, we adopt a subtractive search through the style powerset for C (see below), which is found to be empirically quicker (for instance, if there are more occasions when green indicates grass than not, then a subtractive search makes sense). The full StyP-Boost algorithm can be expressed as in Algorithm 1.

Greedy search through style powerset for T: To avoid searching exhaustively through C^C for the set of shared styles, T, each round begins by setting $T = C$, and greedily subtracts style categories c based on their improvement to J_m until no further improvement is possible.

4 Experimental Results: LabelCracks Database

For our first experiment, we evaluated StyP-Boost on the task of road surface parsing, creating a new database for the task, LabelCracks (see Figure 1). These grey-scale images were captured in strips at a fixed frequency using two downward facing cameras and high intensity strobes, and stitched together into 2148×2675-pixel mosaics. The dataset was prepared by randomly selecting 100 images, down-sampling to 430×535-pixels, and labeling these according to 9 classes (see Figure 1a-d). The main interest is in discriminating cracks for road quality assessment, although the other classes are of general interest for instance in navigation applications. The dataset was split 45/10/45 for training, validation and testing.

	road	gutter	p'ment	kerb	paint	veg'n	drain	crack (t)	crack (p)	overall
DenseBoost	**0.955**	0.621	0.529	0.644	0.617	0.668	0.636	0.371	0.551	0.8863
StyP-Boost (2)	0.949	**0.675**	0.468	0.633	0.626	**0.683**	0.672	0.389	0.536	0.8921
StyP-Boost (3)	0.954	0.584	**0.553**	**0.647**	**0.627**	0.672	**0.686**	0.398	**0.615**	**0.8945**
StyP-Boost (4)	0.949	0.585	0.429	0.619	0.605	0.460	0.476	**0.400**	0.515	0.8909

Table 1: Comparing performance on individual classes of LabelCracks for DenseBoost and StyP-Boost with 2-4 styles using precision and overall proportion correct.

For our baseline we used DenseBoost [7], an extension of TextonBoost which includes quantized HOG features as well as textons (see Section 2). For StyP-Boost, we used these same features. We trained on pixels subsampled regularly across the training set. To provide the style parameters, $\mathbf{p}_i$, we clustered the images into C clusters based on their net HOG-cluster and texton histograms. This was done by using EM to fit a joint mixture of multinomials to the HOG and texton histograms, $Pr(h^{hog}, h^{tex}) = \sum_c \alpha_c \, \text{Mult}(h^{hog}|\theta_c^{hog}) \, \text{Mult}(h^{tex}|\theta_c^{tex})$, where h^{hog} and h^{tex} are the histograms, c ranges across the appearance-clusters, and α, θ are the model parameters. All training instances from the same image share the same style parameters, $\mathbf{p}_i$, whose c'th entry was formed by normalizing $Pr(c|h_i^{hog}, h_i^{tex}) + \beta$. The constant β was found to help sharing data across styles, and was set using the validation data. Style parameters for test images were assigned similarly. 2, 3 and 4 such appearance contexts were used, and 5000 rounds of boosting for all algorithms. Examples of the style categories found using 4 clusters are shown in Figure 1e-h, corresponding to lighting and surface features.

Figure 2 compares training and test performance of the algorithms across 5000 rounds of boosting in terms of proportion of pixels correctly labeled. As shown, StyP-Boost both fits the training data better and achieves better generalization than DenseBoost, with the 3-style classifier performing best overall. Table 1 shows how introducing style parameters helps improve the precision on almost all classes (TP/(TP+FP)), particularly the target classes 'cracked tarmac' and 'cracked paint'. Figure 1a-d gives qualitative comparisons.

5   Experimental Results: Corel Database

We next evaluated StyP-Boost on scene-parsing using the Corel database. This consists of 100 180×120-pixel images of Arctic and African scenes, with 7 object classes (see Figure 3). We created our own train/validation/test split of 45/10/45 images as none is standard.

Algorithm 1 Style Parameterized Boosting (StyP-Boost)

Input : Training instances $\{\mathbf{x}_i, \mathbf{p}_{c,i}, \mathbf{z}_{k,i}\}$
Initialize : $\mathbf{w}_{k,i} = 1$, $H_{1...K} = 0$, $K = $ # of object classes, $C = $ # of style categories
for $m = 1...M$ **do**
    for $v = N$ random features **do**
        for subtractive search through $T \in C^C$, additive search through $S \in K^K$, all θ's **do**
            solve for optimal a, b and $\kappa^{1...K}$ via (8)
            record $J_m = \sum_{k,i} w_{k,i}^m (z_{k,i} - h_m'(k,i))^2$ using current $(v, T, S, \theta, a, b, \kappa^{1...K})$
        end for
    end for
    choose optimal parameters for weak-learner h_m' based on J_m's, and add to $H_{1...K}$
    update weights $\mathbf{w}_{k,i}$ via (3)
end for
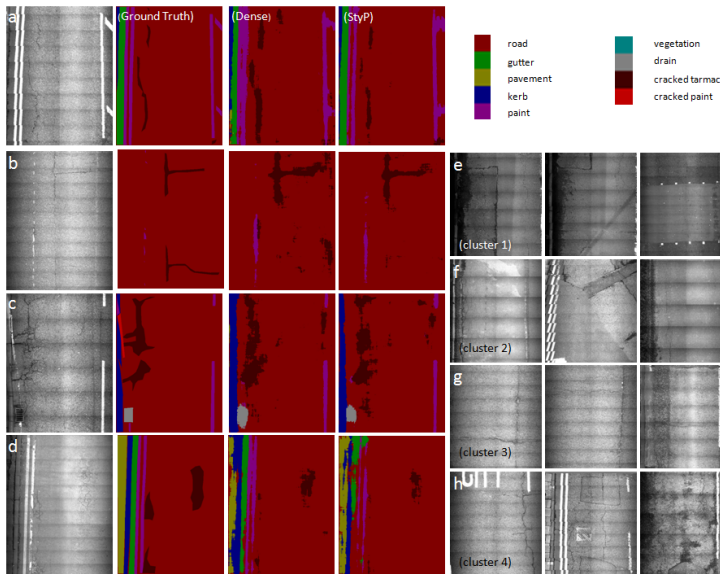**Output** : The learned classifiers $H_{1...K}$

Figure 1: Results for image parsing using the LabelCracks database. (a)-(d) juxtapose the original image, ground truth, DenseBoost and StyP-Boost (3/4-style) results using the key given. Notice that StyP-Boost reduces the false detection of cracks in all images. (e)-(f) give example images from the four global appearance clusters that were used to train StyP-Boost.

	hippo	p'bear	water	snow	ground	veg'n	sky	overall
DenseBoost	0.629	0.646	0.715	0.565	0.595	0.613	**0.561**	0.767
StyP-Boost (2)	0.642	0.681	0.845	0.703	0.644	0.697	0.375	0.822
StyP-Boost (2-4)	0.702	0.713	0.886	0.766	0.645	0.716	0.406	0.845
StyP-Boost (2-4) + CRF	**0.736**	**0.746**	**0.907**	**0.776**	**0.671**	**0.735**	0.392	**0.860**

Table 2: Comparing performance on individual classes of the Corel database for DenseBoost and StyP-Boost (see text) using union-intersection (TP/(TP+FP+FN)) and overall proportion.

We again compare with DenseBoost, using Colour-HOG, HOG and texton features. Here though, we based the style parameters on semantic scene-classes. These were created by clustering the label histograms of the training images: For C clusters, we used EM to fit a mixture of multinomials with C components to these histograms, $Pr(h^{lab}) = \sum_c \alpha_c \text{Mult}(h^{lab}|\theta_c)$, where h^{lab} are the label histograms, c are the scene-classes/styles, and α, θ are the model parameters. For training, we can form the vectors $\mathbf{p}_i$ by setting the c'th entry to $Pr(c|h_i^{lab}) + \beta$ and normalizing, where β is set on the validation data to avoid overfitting. For test data though, we do not know the label histograms. We therefore trained a further DenseBoost classifier to predict scene-class using features across the whole image (i.e. rectangles r defined by their absolute location), using as ground truth the cluster assignments of the training set. We set $p_{ic} \propto \exp(H(c,i)) + \beta$, where $H(c,i)$ is the classifier output for style c. Performance was tested using 2, 3 and 4 styles, 500 rounds of boosting for the scene-class classifier, and 5000 rounds for the overall classifier. Figure 3f-i shows the scene-classes for the 4-style algorithm, revealing clear semantics (the 2-style algorithm clustered f-i and g-h, and the 3-style f-i).
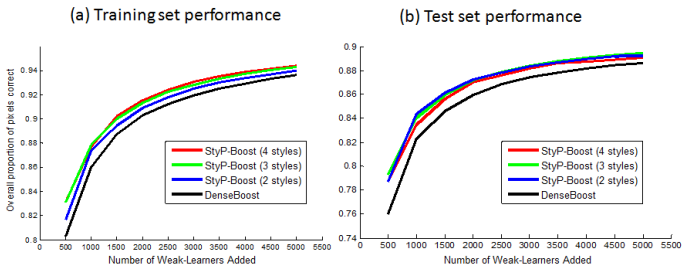
Figure 2: Plot of (a) training set and (b) test set performance across 5000 rounds of boosting for DenseBoost, and StyP-Boost with 2-4 styles. Results are given in terms of proportion of pixels correct on the LabelCracks database.

	DenseBoost	StyP-Boost (2)	StyP-Boost (2-4)	Shotton [▯]	He [▯]	He [▯]	Batra [▯]
unary	0.767	0.822	**0.845**	0.684	0.669	0.710	0.632
unary + CRF	0.787	0.837	**0.860**	0.746	0.800	0.810	0.828

Table 3: Comparing performance against previous results on the Corel database, unary and full CRF results given. Performance given as overall proportion of pixels correctly labeled.

Figure 4 compares performance on training and test data. Figure 4a shows increasing the number of styles fits the training data better, but unexpectedly there is no direct link to generalization in Figure 4b, where the 2-style model does best. As explanation, consider the failure case shown in Figure 3e: if the estimate of scene-class is wrong, we see it drastically decreases performance. Indeed, as the number of scene-classes increases, the accuracy of the scene classifier itself decreases (our scene classifier achieves 0.978, 0.956 and 0.889 accuracy for 2, 3 and 4 styles respectively), which negatively affects overall performance. To counter this effect, Figure 4c plots the performance on the subset of the test set whose scene-class is correctly identified by 2, 3 and 4-style algorithms alike. As shown, more styles *are* beneficial here, implying that if classified correctly, it is preferable to use more styles. Figure 4c also plots the performance of a model in which 2 DenseBoost classifiers are trained separately on the same styles as the StyP-Boost-2 model, and the scene classifier is used to decide which to run (labeled 'DenseBoost × 2'). As expected, the model outperforms a single DenseBoost, but not StyP-Boost(2), indicating that the learning across styles in StyP-Boost is important.

The set used in Figure 4c is of course selected post-hoc. However, the fact that we have a model for the overall scene-class label histogram ($Pr(h^{lab}|c) = \text{Mult}(h^{lab}|\theta_c)$) suggests we may be able to identify failure cases like Figure 3e automatically based on their log-likelihood. We thus propose the following algorithm to combine the StyP-Boost-2, 3 and 4 classifiers: For each image, we begin by running StyP-Boost-4, and evaluate $\log Pr(h^{lab}|c)$ on the result. If this value is above a threshold, we accept the result, and if not we repeat the process using StyP-Boost-3, StyP-Boost-2 and finally DenseBoost, until a segmentation is accepted. The thresholds are set on the validation set to be sufficiently high that all false positives are rejected. This algorithm can be viewed as approximating inference in a larger-scale CRF model which treats scene-class as a latent variable tied to both the unary potentials and a 'whole-scene' clique involving $Pr(h^{lab}|c) \cdot Pr(c)$. Using this approach, we were able to gain improvement on the 2-style algorithm (see Table 2, StyP-Boost(2-4)), while embedding the model in a CRF with a contrast-sensitive edge term improved results further (Table
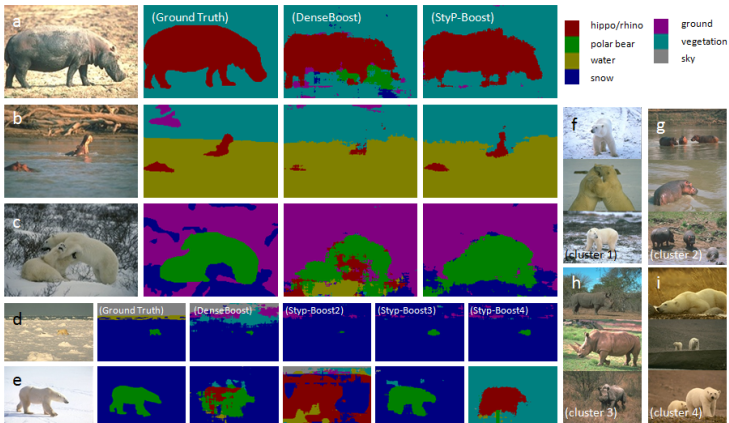
Figure 3: Results for scene parsing using the Corel database. (a)-(c) juxtapose original image, ground truth, DenseBoost and StyP-Boost labelings, where labels are according to the key given, and the 'best' StyP-Boost result is chosen (2, 3 and 4 styles are trained, and a heuristic decides which to choose, see text). Notice that StyP-Boost both eliminates inappropriate classes (a)(c), and gives a stronger signal for true classes (b). (d) shows a case which demonstrates how a weak polar-bear signal improves as more styles are added. (e) shows a failure case, where the misassignment of style parameters in the 2- and 4-style settings massively degrades the results. The 3-style result however is good. (f)-(i) show images from the 4 styles found (Arctic-snow, Africa-hippos, Africa-rhinos, Arctic-vegetation).

2, StyP-Boost(2-4)+CRF). Table 3 compares overall performance against previous results, which show a notable improvement even considering the lack of a fixed training/test split, while Figure 3a-e show qualitative results (see caption).

6 Discussion

We have introduced *Style Parameterized Boosting* to learn a bilinear model in a JointBoost/TextonBoost framework [10, 14], and shown it to offer advantages on two object class labeling tasks. Our method is characterized by its ability to vary the way it discriminates objects with style, while also learning about object classes across styles. One limitation of our approach is that we must fix the style parameters before training. For larger databases such as VOC [2] where there is a less clear notion of style at the image level than the Corel database, it would be preferable for the algorithm to find the most discriminative style parameters itself. This is explored in 'latent aspect' bilinear SVMs in [3], and finding an equivalent boosting formulation would be desirable for parsing applications. Another limitation as discussed above is the difficulty in recovering from mistakes in the algorithm presented, which prompted the heuristic in Section 5. A better approach would be to embed the latent scene-class variable in a larger probabilistic model, which can integrate information across a hierarchy of levels (possibly an associative hierarchical CRF [11], or a grammar-based model [16]). Finally, we note bilinear/multilinear models are under-explored in discriminative frameworks, and other models (e.g. decision trees) may benefit from their application.
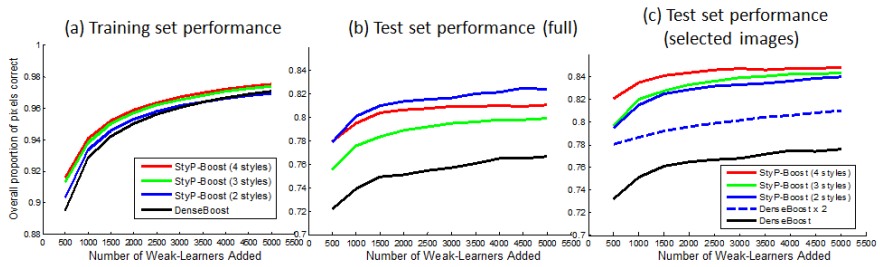
Figure 4: Plot of (a) training set and (b) test set performance across 5000 rounds of boosting for DenseBoost, and StyP-Boost with 2-4 styles using the Corel database. Although more styles implies a better fit on training data (a), this does not necessarily imply better test performance (b). However, compared only on test images whose scene-class is correctly identified on all settings, improved performance is revealed (c). This motivates a combined algorithm, which uses more contexts only if it is sure of the assignments (see text). (c) also compares performance using separate boosting on 2 styles (labeled DenseBoost × 2).

References

[1] D. Batra, R. Sukthankar, and T. Chen. Learning class-specific affinities for image labelling. In *CVPR*, 2008.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PAS-CAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[3] A. Farhadi, M.K. Tabrizi, I. Endres, and D. Forsyth. A latent model of discriminative aspect. In *ICCV*, 2009.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. In *Annals of Statistics*, 2000.

[5] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[6] X. He, R.S. Zemel, and D. Ray. Learning and incorporating top-down cues in image segmentation. In *Lecture Notes in Computer Science*, 2006.

[7] L. Ladicky, C. Russell, P. Kohli, and P.H.S. Torr. Associative hierarchical crfs for object class image segmentation. In *ICCV*, 2009.

[8] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Bilinear classifiers for visual recognition. In *NIPS*, 2009.

[9] J. Shotton, J. Winn, C. Rother, and A Criminisi. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[10] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *IJCV*, 2009.

[11] P. Sturgess, K. Alahari, L. Ladicky, and P.H.S. Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC*, 2009.

[12] J.B. Tenenbaum and W.T. Freeman. Separating style and content with bilinear models. In *Neural Computation*, 2000.

[13] A. Torralba. Contextual priming for object detection. In *IJCV*, 2003.

[14] A. Torralba, K. P. Murphy, and W.T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 29(5):854–869, 2007.

[15] A. Torralba, K. Murphy, and W.T. Freeman. Using the forest to see the trees. In *Communications of the ACM*, 2010.

[16] Z. Tu, X. Chen, A.L. Yuille, and S.C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *IJCV*, 2005.