Active Online Confidence Boosting for Efficient Object Classification

Dennis Mund Rudolph Triebel Daniel Cremers

***Abstract*— We present a novel efficient algorithm for object classification. Our method is based on the active learning framework, in which training and classification are performed in loops, and new ground truth labels are queried from the supervisor in each loop. Our underlying classifier is from the family of boosting methods, but in contrast to earlier methods, our Confidence Boosting particularly focusses on misclassified samples that have a high classification confidence associated. We show that weighting these samples more than others leads to a decrease of overconfidence, for which we give a formal definition. As a result, our classifier is better suited for active learning, leading to steeper learning curves and less required label queries. We show the benefits of our approach on standard data sets from machine learning and robotics.**

I. Introduction

Object classification is one of the most important tasks for a mobile robot, because for many kinds of interactions with the environment or with a human user, the robot needs semantic information about the environment. For that reason, research in this topic is performed by a large community, both within robotics and computer vision, and a number of good approaches have been presented in the past. The focus of these methods, however, differs slightly with the application: While in offline learning run time and memory efficiency are often less important, robotics is often concerned with online learning, because robots need to make fast decisions and update their internal representations with little computational effort. Furthermore, an important aim is to reduce the amount of required user interaction. In the case of object classification this means that the required amount of human-labeled training data should be small. Finally, for a mobile robot it is very advantageous to have a reliable measure of *confidence* along with the classification results, because often important and safety-critical decisions depend on them, and a restriction to very confident classifications can help to avoid accidents, for example in autonomous driving applications [1], [2].

In this paper, we present a novel learning method that addresses all these three issues. Our approach is based on an active learning framework, in which the human user is involved in the learning process, and learning is done in cycles of iterated training and inference. As we will show experimentally, this reduces the amount of required hand-labeled training data, and at the same time increases the classification performance. As an underlying classification method we use a novel online multi-class boosting algorithm. The motivation to choose this classifier stems from the very fast computation time used for training and inference, and from the online nature of the algorithm, i.e. it can update its internal representation and make predictions before having seen all input data. This is particularly useful in active learning, because it reduces the run time. However, as we will show, standard online boosting methods (e.g. Saffari *et al.* [4]), are not very useful for active learning, because they tend to associate wrong classifications with a too large confidence, i.e. they are often *overconfident*. This has a severe effect, because it prevents the classifier from finding misclassified samples and, as a result, reduces the chance to obtain ground truth labels from the human for re-learning these erroneously classified samples. An illustrative example is given in Fig. 1. Here, the input image shows a light bulb, but the available information is too low to classify this object correctly. Both standard gradient boosting and our Confidence Boosting method report a wrong class label. However, after 10 rounds of active learning (on a data set that is different from this test data), Confidence Boosting is able to classify the object correctly, while gradient boosting is not. In our experiments, we will show that this is due to the reduced overconfidence of our classification algorithm.

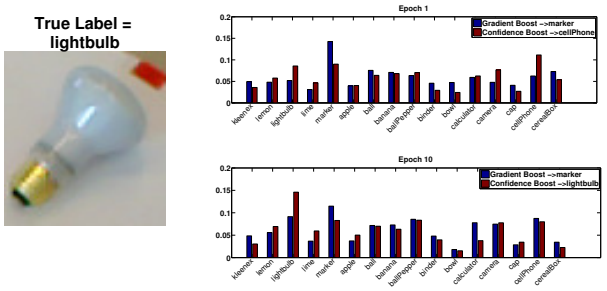

Fig. 1: Object classification with active online Confidence Boosting: The image on the left was recorded with an RGB-D sensor [3]. On the right, we show class label predictions obtained from online gradient boosting [4] and our online Confidence Boosting. The upper row shows results after one learning round, the lower row depicts results after 10 rounds of active learning. As we can see, both classifiers first return a wrong class label (see arrows in legend), but Confidence Boosting can recover from the error and finally give a correct label. In this paper, we show that this is because Confidence Boosting is less overconfident.

A. Related Work

The existing literature on object classification is too large to be mentioned exhaustively here. Therefore we only name some of the most recent achievements. These include deep belief nets [5], convolutional deep belief networks (CDBN) [6] and fully connected Conditional Random Fields (CRFs) [7]. Despite their impressive results, these classifiers focus

All authors are with Computer Vision Group, Department of Computer Science, Technische Universität München, Germany {dennis.mund,rudolph.triebel,daniel.cremers}@in.tum.de

only on the classification rate, whereas we are are also interested in the overconfidence of a classifier. Furthermore, sparse coding techniques have become popular, and in particular the Hierarchical Matching Pursuit (HMP) algorithm [3], which we also use to compute descriptors, however with a classifier that is superior in performance compared to the linear support vector machine (SVM) used there.

Our approach is formulated as an active learning method. This research area is experiencing a growing interest in the area of computer vision and robotics. For example, Kapoor *et al.* [8] use active learning for object categorization using a Gaussian Process classifier (GPC) where labels are queried for data points that are classified with high uncertainty. Triebel *et al.* [9] use a sparse version of the GPC, the Informative Vector Machine (IVM) to actively learn traffic lights in urban traffic images. In computer vision, Vezhnevets *et al.* [10], as well as Wang *et al.* [11] use active learning for interactive image segmentation. In contrast to all these methods, we propose to use a boosting method as an underlying classifier, because it is more efficient in terms of training and evaluation time. In that context, a work that is very closely related to ours is that of Saffari *et al.* [4], which proposes an efficient multi-class online boosting algorithm. We extend that approach using a similar idea as presented in [12], with the difference that here we use it for online boosting and with more theoretical justifications. Concretely, we introduce a formal definition of over- and underconfidence of a classifier on a given data set. This is related to the intuitive notion of *introspection* introduced by Grimmett *et al.* [1], however with the difference that our formulation can explicitly quantify the inherent trade-off between a high number of detected misclassifications and a high number of correct classifications that are not further used for training.

II. Active Learning

The main difference between standard passive learning and active learning is that, instead of strictly separating between a *training* and a *testing* phase, the active learner performs *loops* of training and testing, thereby incorporating the information flow obtained from the teacher (e.g. a human supervisor) into the loop. Fig. 2 shows a schematic flow chart of a generic active learning algorithm. We note that, while in general active learning can be used in many different contexts, we will use it for object classification in this work. In the following, we will describe the indivudal components of our framework in more detail.

A. Components of Active Learning

Similar to standard passive learning methods, in active learning we start with an initial training set $(\mathcal{X}_0, \mathcal{Y}_0)$, where $\mathcal{X}_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ are the N feature vectors with d dimensions, i.e. $\mathbf{x}_i \in \mathbb{R}^d$, and $\mathcal{Y}_0 = \{y_1, \ldots, y_N\}$ are the corresponding class labels, i.e. $y_i \in \{1, \ldots, C\}$. In this paper, we assume the number of classes C as given, but usually larger than 2. We note that, in contrast to passive learning, active learners usually can deal with much smaller initial

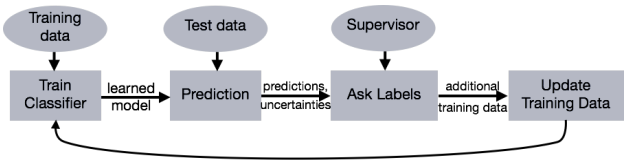

Fig. 2: Active Learning flow chart. After an initial training step, the classifier is presented new test data and reports label predictions and confidence values (here: uncertainties). These are used to ask a human supervisor for new ground truth labels, which subsequently are added to the current training data. Then, the training process is repeated with the extended training data until a stopping criterion is met.

training data sets, which is one major advantage of active learning methods. The first step is then to train a classifier with the initial training set, which we model as a function $f : \mathbb{R}^d \to \mathbb{R}^C$, i.e. each input feature vector $\mathbf{x}$ is mapped to a prediction vector $\mathbf{p} \in \mathbb{R}^C$. We will give more details on this in Sec. III. Next, a set of K data samples $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_K^*\}$ is drawn from the test data set $\mathcal{X}^*$ and classified using f. Here, the nature of $\mathcal{X}^*$ defines the type and complexity of the active learning problem: if $\mathcal{X}^*$ is given beforehand and its size does not change during learning, then we are concerned with *pool-based* active learning. If, however $\mathcal{X}^*$ is a potentially infinite stream of data, then we have *stream-based* active learning, which is significantly harder. In our work, we consider the pool-based variant, although we aim to extend the approach to stream-based learning in the future.

The result of the classification of $\{\mathbf{x}_1^*, \ldots, \mathbf{x}_K^*\}$ are the *label predictions* $y_1^*, \ldots, y_K^*$, where

$$y_k^* = \arg\max_c(p_1, \ldots, p_C), \qquad \mathbf{p} = f(\mathbf{x}_k^*). \tag{1}$$

In addition to the label predictions, the prediction vectors themselves also play an important role, because they can be used to distinguish test points, where the classifier has a high chance of correct classification from those where the classification is likely to be incorrect. We refer to this as the *confidence* of the classifier. This can be computed based on the *uncertainty* of the classification, and we will give more details below. Now, the key element of active learning is the ability of the learner to *query* new class labels from the human supervisor. This is usually done by selecting those test points $\mathbf{x}_i^*$, for which the classifier has a low confidence and asking a ground truth label $\hat{y}_i$ for them. Here, we note that the number K of new test samples considered can be used to bound the required human effort by querying only the least confident ones. For $K = 1$ this would result in a query for every sample, but for larger K the benefit becomes obvious. Then, after the label query, the new data-label pairs $(\mathbf{x}_i^*, \hat{y}_i)$ are added to the current training data $(\mathcal{X}_{j-1}, \mathcal{Y}_{j-1})$, where j is the index of the current learning round, and the learning process starts again until an appropriate stopping criterion is reached. In our implementation, we use a fixed number of learning rounds.

B. The Which-Question Problem

One important question in active learning is how to select the data samples for which semantic information, i.e. in our

case class labels, are requested from the human supervisor. We refer to this as the *which-question problem*. In the survey of Settles [13] the following *query strategies* are summarized to address this problem: uncertainty sampling, query-by-committee, expected model change, expected error reduction, variance reduction, and density weighting. Among these, the most used method is the uncertainty sampling, and we also use it in our implementation. Assume that the entries of the prediction vector $\mathbf{p}$ returned by f for a test sample $\mathbf{x}^*$ sum up to 1, i.e. $\sum_{i=1}^{C} p_i = 1$. Then, each entry p_i in $\mathbf{p}$ can be interpreted as the probability that $\mathbf{x}^*$ has label i. From this, there are two common ways to compute uncertainty:

$$h_e(\mathbf{p}) := -\sum_{i=1}^{C} p_i \log_C(p_i) \qquad h_b(\mathbf{p}) := p_{i_2}/p_{i_1}, \tag{2}$$

where i_1 and i_2 are the indices of the largest and second largest values of $\mathbf{p}$, respectively. The first measure is the *normalized entropy*, where the base of the logarithm is C so that h_e always ranges between 0 and 1, and the second is a variant of the *best-vs-second-best* (BVSB) method. Both measures are usually very well suited for active learning, while BVSB tends to perform slightly better for multi-class classification tasks such as ours. With these definitions of uncertainty h, we define the classification confidence as $1-h$, and we will use both terms in the following.

Now, to address the which-question problem, the standard uncertainty sampling approach uses a *confidence threshold* ϑ_c and decides to ask for a ground truth label $\hat{y}$ for all those data samples which, in the current learning epoch, have been classified with a confidence lower than ϑ_c. This directly raises two questions: What is a good choice for ϑ_c? And how do we know that the classifier gives meaningful uncertainty estimates so that uncertainty sampling actually makes sense? While the first question will be answered in Sec. III-C, the second one will be addressed next.

C. Under- and Overconfidence

A crucial point with the uncertainty estimates obtained from the classification is the question, how much one can rely on these estimates. Formally, our aim is to have a high correlation between prediction uncertainty and incorrectness of the classification (a similar idea was used by Zhang et al. [14]). In [12], this correlation was measured using the point-biserial correlation coefficient. However, this turns out to be too restrictive and only applicable in cases where the number of correctly and incorrectly classified samples is roughly balanced. Therefore, we take a different approach. For a test set $\mathcal{X}^*$ of size K we define two functions u and o as follows:

$$u(f, \mathcal{X}^*, \hat{\mathcal{Y}}) := \frac{1}{K_c} \sum_{\mathbf{x}^* \in \mathcal{X}^*} I(y^* = \hat{y}) h(f(\mathbf{x}^*)) \tag{3}$$

$$o(f, \mathcal{X}^*, \hat{\mathcal{Y}}) := \frac{1}{K_f} \sum_{\mathbf{x}^* \in \mathcal{X}^*} I(y^* \neq \hat{y})(1 - h(f(\mathbf{x}^*))), \tag{4}$$

where I is the indicator function, $\hat{\mathcal{Y}}$ are the ground truth labels, and K_f and K_c are the number of incorrectly and correctly classified test samples, i.e. $K_f + K_c = K$. Thus, u is the average *uncertainty* of the *correct* classified samples and o is the average *confidence* of the *incorrectly* classified samples. We will denote u as the *underconfidence* and o as the *overconfidence* of the classifier. Intuitively, if all incorrect classified samples have the maximum confidence of 1 assigned, then the overconfidence reaches its maximum value of 1. This is the worst case for active learning, because the classifier is unable to give an indication that its predicted class label is wrong. As a consequence, the algorithm will never ask ground truth labels for the incorrectly classified samples, and they can not be used for re-training. Thus, the classification can not be improved.

Another extreme case is that of maximal *underconfidence* u. Here, all uncertainty values h for correctly classified samples are 1, i.e. the classifier is always fully uncertain, although the corresponding predictions are correct. The problem with this case is that active learning will be very inefficient, because very often the algorithm queries ground truth labels for samples that are already correctly classified. It is important to note that underconfidence and overconfidence are in this definition completely independent quantities. In particular, a classifier can be under- and overconfident at the same time, namely when it is uncertain on the correct predictions and certain on the wrong ones.

Despite the problems with inefficiency caused by under-confident classifiers, we will focus more on the task to avoid overconfidence, as this has the more severe effect on active learning. However, the problem here is that we can not explicitly minimize o using a closed-form expression. Also, we have to make sure that the overconfidence is reduced *simultaneously* with the reduction of the training error. To do this, we propose to extend a standard online multi-class boosting algorithm in such a way that it also takes classification confidences into account. This will be described next.

III. Online Confidence Boosting

In principle, there are two different ways to achieve classification results with little overconfidence: either we use a classifier that is already known to be less overconfident than others, or we modify an existing algorithm so that it is less overconfident. If we follow the first idea, then a good choice for a classifier is the Gaussian Process classifier (GPC), as was shown earlier [1], [15], because due to its capability to marginalize over a range of potential models, its uncertainty estimates are more reliable because they correlate more with actual misclassifications (see [1] for more details). One major problem however, is its huge demand in run time and memory. Even though there are sparse and more efficient variants such as the Informative Vector Machine (IVM) [16], the method is still hardly applicable for typical data sets in mobile robotics. Furthermore, as we are investigating active learning here, we have even stricter requirements on the run time, because the user is involved in the learning process, and learning should be done during operation of the robot.

Algorithm 1: Online Multi-class Gradient Boost [4]

Data: training data $(\mathcal{X}, \mathbf{y})$ with C classes
Input: number of weak learners M, loss function ℓ, agreement function a
Output: weak learners $f_1, \ldots, f_M$

1 `Initialize`$(f_1, \ldots, f_M)$
2 **for** $n = 1, \ldots, N$ **do**
3 $\quad w_n \leftarrow 1$
4 $\quad g_n \leftarrow 0$
5 $\quad$ **for** $m = 1, \ldots, M$ **do**
6 $\quad\quad f_m \leftarrow$ `UpdateWeakLearner`$(f_m, \mathbf{x}_n, y_n, w_n)$
7 $\quad\quad \mathbf{p}_{nm} \leftarrow f_m(\mathbf{x}_n)$
8 $\quad\quad \alpha_{nm} \leftarrow a(\mathbf{p}_{nm}, y_n)$
9 $\quad\quad g_n \leftarrow g_n + \alpha_{nm}$
10 $\quad\quad w_n \leftarrow -\nabla \ell(g_n)$

Therefore, we decided to use a classification framework that is known to be efficient and effective, and that can be modified so that it is less overconfident. A recently developed method with these requirements is the Online Multi-Class Gradient Boost (OMCGB) algorithm of Saffari *et al.* [4] (see Algorithm 1), which we describe next.

A. Online Multi-Class Gradient Boost

In addition to the training data, the OMCGB algorithm requires three different parameters as input: a fixed number M of weak learners, a loss function $\ell : \mathbb{R} \to \mathbb{R}$, and an *agreement function* $a : \mathbb{R}^C \times \mathbb{N} \to \mathbb{R}$, which quantifies the amount of agreement between a class label prediction $f(\mathbf{x}_n)$ and the corresponding ground truth label y_n. After initialization of the weak classifiers, the algorithm loops over all training data points and updates all weak classifiers for every new training sample $(\mathbf{x}_n, y_n)$. This online behaviour of the algorithm is very attractive for our active learning framework, because it avoids a recomputation of the underlying representation whenever a new ground truth label is queried from the user and added to the existing training set. As in offline boosting methods, every training sample $\mathbf{x}_n$ has an assigned weight w_n, which is first initialized to 1. Then, every weak classifier is first updated with the new sample, its weight w_n and its ground truth label y_n. Note that the weak learner itself also must be an online algorithm, because otherwise the overall boosting method would not be online. In our experiments we used online random forests as weak classifiers.

The next step (line 7) is to obtain a label prediction $\mathbf{p}_{nm}$ for the new training sample. Then, the agreement with the ground truth label is computed. In standard OMCGB, this is defined as

$$a_g(\mathbf{p}_{nm}, y_n) = p_{nm}^{(y_n)} - 1/C, \tag{5}$$

i.e. it is directly related to the prediction for class y_n, here denoted as an upper index into the prediction vector $\mathbf{p}_{nm}$. The resulting agreement α_{nm} is then accumulated, and a new weight w_n is computed for the sample from the negative gradient of the loss function of the sum of agreements. In [4], two different loss functions are investigated, but with little performance difference, so we decided to use the standard exponential loss $\ell(g) = \exp(-g)$ known from AdaBoost. Concretely, the computation in line 10 results in higher weights for samples that disagree with the ground truth and lower weights for those that do agree.

B. Extension to Confidence Boosting

As can be seen from Eq. (5), the agreement a_g used by standard gradient boost is only related to the prediction itself, but not to the confidence of the prediction. To build a classifier that takes both prediction and confidence into account, we propose to use this agreement function:

$$a_c(\mathbf{p}_{nm}, y_n) = (-1)^{\xi} \left(1 - \frac{h(\mathbf{p}_{nm})}{(C-1)^{\xi}}\right), \tag{6}$$

$$\text{where } \xi = I(\arg\max_i \mathbf{p}_{nm}^{(i)} \neq y_n). \tag{7}$$

This means, that we measure agreement by the amount of confidence, which is equal to one minus uncertainty. In case of a correct classification, i.e. when $\xi = 0$, the agreement simply amounts to the confidence of the current weak classifier f_m. However, if the classification is incorrect, we actually have a disagreement, and we express this with the – slightly modified – *negative* confidence. Our modification is the term $(C-1)$, by which we divide the uncertainty. This has empirically shown to improve the classification results substantially. To summarize, our agreement function is high if the classification is correct and certain, and it is low if we have an incorrect, but certain classification. Also note that if the uncertainty is zero, i.e. when we completely trust the classification, then the agreement is 1 for correct and -1 for incorrectly classified samples. Thus, in this case, our agreement function is even simpler than the original one given in Eq. (5).

C. Adaptive Thresholding

As mentioned in Sec. II-B, active learning with uncertainty sampling requires to specify the confidence threshold ϑ_c to decide for which samples a ground truth label should be queried. Usually, this threshold is a fixed parameter of the algorithm that does not change during the learning process. However, apart from the fact that it is in general difficult to find a good value for ϑ_c, having a fixed value often leads to a poor performance either in terms of efficiency or in terms of classification rate. The reason is that there is an inherent trade-off in the choice of ϑ_c. If it is small, then the number of label queries is low, thus increasing efficiency in the next learning round, but with a potentially lower classification rate. In contrast, if ϑ_c is large we have a higher chance of finding those samples, for which the classifier was wrong, which is good to correct for misclassifications, but it also increases the risk of re-learning already correctly classified samples. In addition to this, ϑ_c should also be chosen according to the level of over- and underconfidence of the classifier. For example, if the classifier tends to be overconfident, then a higher value of ϑ_c should be used to increase the chances to find misclassifications. To address

this issue, in our implementation we use an adaptive method: In every training epoch we compute confidence histograms for correctly and incorrectly classified samples from the previous epochs. Then we start a search at $\vartheta_c = 0$ with a positive step size until the fraction of false samples with a confidence below ϑ_c equals the fraction of correct samples with a confidence above ϑ_c. As we will see later in the experiments, this method provides a good compromise, and it uses a similar idea than the equal-error-rate in a precision-recall graph.

IV. Experimental Results

To evaluate our algorithm, we performed three different experiments on six different data sets. The first experiment investigates how much Confidence Boosting actually reduces the overconfidence in the class label predictions. In the second, we analyze the impact of Confidence Boosting within the active learning framework. And in the last experiment, we compare the learning curve and the run time of Confidence Boosting with those of a Gaussian Process Classifier, which is known to perform particularly well in active learning. All data sets and experiments are described in more detail next.

A. Data Sets

We used four data sets from the UCI machine learning repository, and two sets from robotics. The UCI data sets are 'USPS', 'Pendigits', 'Letter', and 'DNA'. We used these because they were also used for evaluation by Saffari *et al.* [4], and our aim is to compare Confidence Boosting with gradient boosting. The robotics data sets we used were an RGB-D set provided by Lai *et al.* [17], and the 3D point cloud data from Paul *et al.* [15]. From the first one, for which we use the identifier 'RGBD', we extracted 89 pre-segmented objects of 17 object classes, resulting in a total of 58372 RGB-D images. Then, we computed Hierarchical Matching Pursuit (HMP) descriptors [3] on the depth channel. The dictionary needed for the HMP features was learned on 5 classes out of 17, mainly for memory reasons. Then, the data was split into a training set of 90% of the data and an evaluation set of the remaining 10 %. The other robotics data set, which we denote as 'Begbroke', consists of 3D point clouds from a car park with 6 classes. The data was segmented automatically, and features were computed for each segment (see [15]). In total, there were 1496 segments, out of which we took 1000 for training and the rest for evaluation. We used this data to be able to compare with the multi-class GP classifier used in [15], both in terms of run-time and classification performance.

B. Reducing Overconfidence

To measure how much Confidence Boosting actually reduces overconfidence, we computed histograms over the confidences for correctly and incorrectly classified samples, both for gradient boosting and for Confidence Boosting. Fig. 3 shows the resulting histograms for gradient boosting (GB) and Confidence Boosting (CB) on the 'DNA' data set. The plots on the left show the confidence histograms

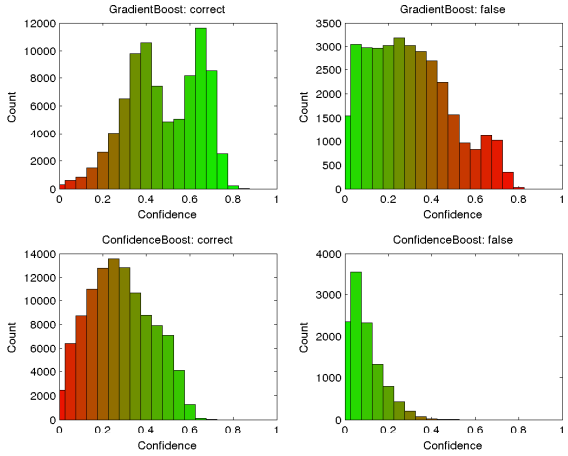

Fig. 3: Confidence histograms of gradient boosting and Confidence Boosting on the 'DNA' dataset. The left plots show the histograms for the correctly classified samples, the right ones show the histograms for the incorrect samples. Confidence Boosting shifts all histograms to the left, resulting in an overall decreased confidence. Note however that the false samples are shifted even further, which leads to a lower overconfidence.

	RGBD	Begbroke	USPS	Letter	Pendigits	DNA
pGB	0.1326	0.3106	0.1978	0.1854	0.2804	0.1475
aGB	0.1064	0.2391	0.1653	0.1395	0.2192	0.1258
pCB	0.0981	0.1715	0.1624	0.1416	0.1682	0.0895
aCB	0.0960	0.1629	0.1515	0.1338	0.1684	0.0871

TABLE I: Overconfidence averaged over 100 runs. We compare passive and active Gradient Boost (pGB and aGB) with passive and active Confidence Boost (pCP and aCB).

for the correct classified samples, the right ones for the incorrect samples (red bars depict either over- or under-confident regions). As we can see, Confidence Boosting tends to shift both histograms to the left, which means that in general classification is more uncertain. This implies that more false classifications are uncertain as well. Thus, increasing the uncertainty in general reduces overconfidence, but it also increases underconfidence. However, as we can see, Confidence Boosting shofts the histograms for the false samples more than the correct ones. A quantitative result is shown in Table I. Note that, e.g. for the 'DNA' data set, the overconfidence is lower both for passive and for active learning when using Confidence Boosting. Although this does not hold for all data sets (see, e.g. 'USPS'), one can say that in general the tendency is that Confidence Boosting reduces the overconfidence.

To visualize the trade-off between over- and underconfidence, we use a plot type similar to a precision-recall curve. On the x-axis, we plot for a given number of different confidence thresholds $\theta_1, \theta_2, \ldots$ the fraction of false classified samples that have a confidence below the thresholds. On the y-axis, we plot for the same thresholds the fraction of correct classified samples for which the classification was more confident than θ_i. Ideally, this curve stays close towards the upper right corner of the plot. Fig. 4 shows an example of such a plot for the 'Pendigits' data set, both for gradient boosting and for Confidence Boosting. We see that Confidence Boosting gives the opportunity to

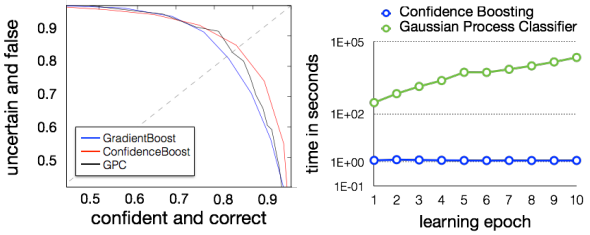

Fig. 4: **Left:** Trade-off curves for gradient boosting, Confidence Boosting, and the GPC on the 'Pendigits' data set. Higher curves correspond to less overconfident classifiers, curves that are further to the right represent a lower underconfidence. We see that Confidence Boosting generally improves over- and underconfidence compared to gradient boosting, and it is even less underconfident than the GPC. **Right:** Average run times of Confidence Boosting (blue) and the GPC (red) for each epoch (note the log scale).

	RGBD	Begbroke	USPS	Letter	Pendigits	DNA
pGB	0.2704	0.2668	0.1536	0.2628	0.1589	0.1410
aGB	0.1824	0.1463	0.1123	0.1876	0.0983	0.0867
pCB	0.1248	0.0568	0.0898	0.1161	0.0559	0.0811
aCB	0.1165	0.0517	0.0726	0.0840	0.0341	0.0724

TABLE II: Average classification errors over 100 runs.

'detect' more false classifications while at the same time not loosing too many correct classifications. The plot also shows how to choose a good confidence threshold ϑ_c for active learning. While our adaptive method chooses the one where the false and the correct classification rates are equal, one could focus more on efficiency by not loosing many correct classifications (towards the right part of the graph) or on classification quality by including more false classified samples into the training set (towards the upper part).

C. Impact of Confidence Boosting for Active Learning

To quantify the effect of Confidence Boosting in the active learning framework, we ran active learning on all 6 data sets, once with standard gradient boosting and once with Confidence Boosting. We repeated this 100 times with the training sets randomly shuffled to obtain results that are independent on the data ordering. The mean classification errors are given in Table II. Apart from the fact that active learning performs better on all data sets than passive learning, where training data was selected randomly and not based on its confidence, we see that Confidence Boosting results in lower classification errors. In particular, Confidence Boosting with active learning performs best on all data sets.

The progress of the learning process for 'Pendigits' is depicted in Fig. 5 (left). Here, the means and standard deviations of the classification error are shown as a function of the generated label queries. We see that Confidence Boosting leads to better results and requires less label queries at the same time. For comparison we also show results on passive learning where the training data was as large as the one for active learning was at the end. Thus, even though passive learning uses the same amount of data, the active learner is better after some learning epochs. Fig. 5 (center) explicitly shows the number of label queries per epoch for three different data sets. Again, Confidence Boosting produces less queries than Gradient Boosting. We relate this to the fact that

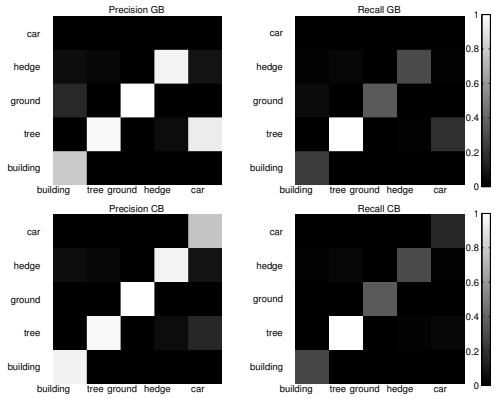

Fig. 7: Normalized confusion matrices for active gradient boosting (GB) and Confidence Boosting (CB) on the 'Begbroke' data set (see text for details).

Confidence Boosting results in a higher correlation between uncertain and incorrect classified samples, which enables the classifier to find more incorrect samples for re-training while at the same time not loosing many correct classified samples. The right part of Fig. 5 shows a comparison of our results from Table II with those reported in [4] and [17]. Note that [4] perform learning on the same data for a number of rounds, which can not be compared to our active learning epochs. Therefore we only compare to the first round reported in [4]. Some qualitative results of our approach are shown in Fig. 6.

D. Comparison to the GPC

In [1], [15] it was shown that a GPC tends to be much less overconfident than other classifiers such as a Support Vector Machine (SVM) or LogitBoost. However, here we show that Confidence Boosting mitigates this issue at least for boosting methods, and we relate that not only to the fact that the confidence is used to compute the sample weights, but also to the choice of the weak classifiers, namely random forests. We argue that with this combination we have a very efficient classification framework that also reduces overconfidence in a way that it comes close to that of a GPC. To back-up this claim quantitatively, we ran a multi-class GPC on the same data sets using an own implementation based on [18]. Unfortunately, for most data sets GPC training could not be done efficiently, therefore we only report results on the 'DNA' data set. Fig. 4 (left) shows that the GPC is only slightly less overconfident than Confidence Boosting. The classification error of the GPC for 'DNA' after 10 epochs was 0.0552, which is slightly better than active Confidence Boosting (see Table II), but at the cost of a much higher run time (see Fig. 4, right).

We also compare our results on the 'Begbroke' data with those reported in [15]. For that, we computed the same normalized confusion matrices as in [15], although without the underrepresented 'Background' class (see Fig. 7). The figure shows that our active Confidence Boosting method reaches a classification performance that is almost as good as that of the GPC, but with a much faster computation time.

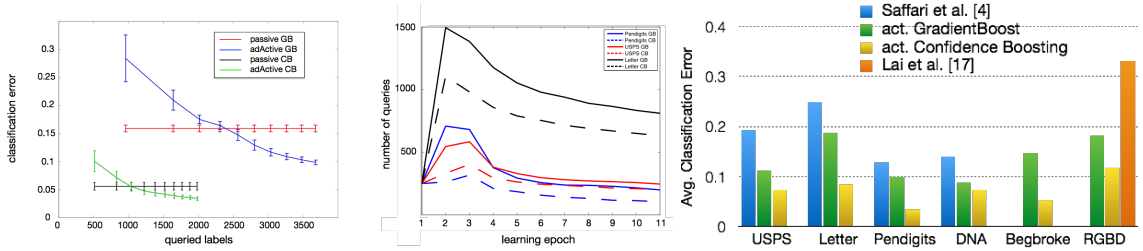

Fig. 5: **Left:** Learning curves for the 'Pendigits' data set. The x-axis shows the number of label queries generated by active learning. Both Gradient Boosting and Confidence Boosting improve with more samples, but Confidence Boosting reaches lower errors and requires less label queries. For comparison, the passive counter parts are shown where the same number of data points was used for training as the active learner has at the end. **Center:** Number of new label queries used per epoch for 'Pendigits', 'USPS' and 'Letter'. All curves start with a pool of 250 samples. We see that in each epoch less additional queries are needed than in the previous one and that Confidence Boosting needs less label queries than gradient boosting. **Right:** Comparison of active learning with the results reported in the literature. Active Confidence Boosting always performs best.

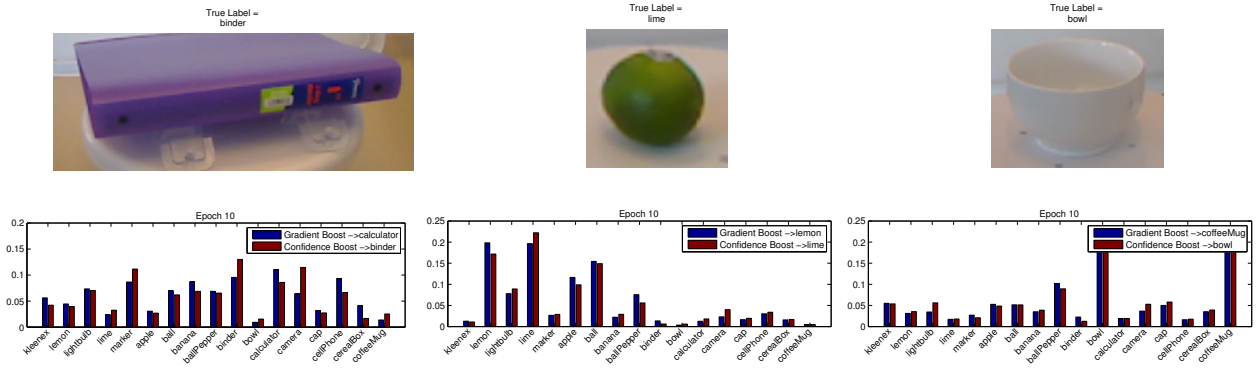

Fig. 6: Qualitative results of our experiments. We show three different objects from the RGBD data set, for which we computed HMP descriptors on the depth information, i.e. colour is not used for classification. After 10 rounds of active learning, Confidence Boosting (CB) returned the correct label, while gradient boosting did not. Note that CB even distinguishes the lime correctly from a lemon although there was no colour information used, i.e. even such small differences in shape can be detected with our approach.

V. Conclusions

Object classification still remains a difficult problem. However, active learning seems to be a very useful technique to tackle this problem, even though it raises the questions of training efficiency and low overconfidence of the classifier. While the latter can be handeled well using a Gaussian Process classifier, this same method is often too slow for online applications. In contrast, our proposed extension of online multi-class gradient boosting is at the same time very efficient and reduces overconfidence over standard boosting, coming close to that of the GPC. The result is an efficient and high performing active learning method, which gives good results even on challenging state-of-the-art data in robotics.

References

[1] H. Grimmett, R. Paul, R. Triebel, and I. Posner, "Knowing when we don't know: Introspective classification for mission-critical decision making," in *ICRA*, 2013.

[2] P. Lamon, C. Stachniss, R. Triebel, P. Pfaff, C. Plagemann, G. Grisetti, S. Kolski, W. Burgard, and R. Siegwart, "Mapping with an autonomous car," in *IEEE/RSJ IROS Workshop: Safe Navigation in Open and Dynamic Environments*, 2006.

[3] L. Bo, X. Ren, and D. Fox, "Hierarchical matching pursuit for image classification: Architecture and fast algorithms," in *Advances in neural information processing systems*, 2011, pp. 2115–2123.

[4] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof, "Online multi-class lpboost," in *CVPR*, 2010.

[5] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, 2006.

[6] H. Lee, R. Grosse, R. Ranganath, and A. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *International Conference on Machine Learning (ICML)*, 2009.

[7] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Conf. on Neural Information Processins Systems (NIPS)*, 2011.

[8] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Gaussian processes for object categorization," *Intern. Journal of Computer Vision*, vol. 88, no. 2, pp. 169–188, 2010.

[9] R. Triebel, H. Grimmett, R. Paul, and I. Posner, "Driven learning for driving: How introspection improves semantic mapping," in *Proc of Intern. Symposium on Robotics Research (ISRR)*, 2013.

[10] A. Vezhnevets, J. Buhmann, and V. Ferrari, "Active learning for semantic segmentation with expected change," in *CVPR*, 2012.

[11] D. Wang, C. Yan, S. Shan, and X. Chen, "Active learning for interactive segmentation with expected confidence change," in *Asian Conf. on Computer Vision*, 2012.

[12] R. Triebel, H. Grimmett, and I. Posner, "Confidence boosting: Improving the introspectiveness of a boosted classifier for efficient learning," in *Autonomous Learning Workshop at ICRA*, 2013.

[13] B. Settles, *Active Learning*. Morgan & Claypool, 2012.

[14] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh, "Predicting failures of vision systems," in *CVPR*, 2014.

[15] R. Paul, R. Triebel, D. Rus, and P. Newman, "Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification," in *IROS*, 2012.

[16] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse gaussian process methods: The informative vector machine," *Adv. in neural inf. proc. systems*, vol. 15, pp. 609–616, 2002.

[17] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *ICRA*, 2011.

[18] J. Riihimäki, P. Jylänki, and A. Vehtari, "Nested expectation propagation for gaussian process classification with a multinomial probit likelihood," *Journal of Machine Learning Research*, vol. 14, 2013.